WEST

End of Result Set

Generate Collection | Print

L35: Entry 1 of 1 File: USPT Jun 6, 2000

US-PAT-NO: 6073139
DOCUMENT-IDENTIFIER: US 6073139 A

TITLE: Integrated data communication and data access system including the application data interface

DATE-ISSUED: June 6, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Jain; Pradeep	Sugar Land	TX		
Ahmed; Shamim	Houston	TX		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
GioQuest, a division of Schlumberger Technology Corp.	Houston	TX			02

APPL-NO: 08/ 848205 [PALM]
DATE FILED: April 30, 1997

PARENT-CASE:
CROSS REFERENCE TO RELATED APPLICATIONS This specification is a Continuation-in-Part application of prior pending application Ser. No. 08/758,833 filed Dec. 4, 1996 and entitled "Distributed Framework for Intertask Communication Between Workstation Applications", and this specification is also a 35 USC 119(e) (1) application of prior pending provisional application Ser. No. 60/023,945 filed Aug. 19, 1996 and entitled "Distributed Framework for Inter--Process Communication Between Workstation Applications", and this specification is also a 35 USC 119(e) (1) application of prior pending provisional application Ser. No. 60/023,689 filed Aug. 15, 1996 and entitled "Application Data Interface (ADI)".

INT-CL: [07] G06 F 17/30

US-CL-ISSUED: 707/203; 707/103, 709/216
US-CL-CURRENT: 707/203; 709/216

FIELD-OF-SEARCH: 707/200, 707/8, 707/203, 707/103, 711/133, 711/135, 711/138, 711/142, 711/159, 711/144, 709/203, 709/204, 709/216

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected | Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4713751	December 1987	Dutton et al.	
☐	5050104	September 1991	Heyen et al.	395/514
☐	5062037	October 1991	Shorter et al.	364/200
☐	5287496	February 1994	Chen et al.	707/203
☐	5524253	June 1996	Pham et al.	395/200.32
☐	5557798	September 1996	Skeen et al.	705/35
☐	5574917	November 1996	Good et al.	
☐	5692187	November 1997	Goldman et al.	707/203
☐	5742778	April 1998	Hao et al.	345/332
☐	5752159	May 1998	Faust et al.	455/5.1
☐	5758149	May 1998	Bierma et al.	707/8
☐	5761500	June 1998	Gallant et al.	707/10
☐	5787280	July 1998	Joseph et al.	707/203
☐	5806075	September 1998	Jain et al.	707/201
☐	5822529	October 1998	Kawai	395/200.49
☐	5826253	October 1998	Bredenberg	707/2
☐	5881292	March 1999	Sigal et al.	395/712

OTHER PUBLICATIONS

Microsoft Corp. "The Component Object Model Specification", pp 1-13. 1995.

ART-UNIT: 271

PRIMARY-EXAMINER: Black; Thomas G.

ASSISTANT-EXAMINER: Le; Uyen

ABSTRACT:

An integrated data communication and data access system includes a first cache memory operatively connected to a first application and a first conversion means operatively connected to the first cache memory and interfacing between a first operator of the first application and the first cache memory for receiving original data having a first format from the first operator and converting the original data having the first format into original data having a second format, the first application storing the original data having the second format in the first cache memory and in a database when the first application sets a persistant or a transient storage state. A second application independently inquires about the existance of the original data having the second format, independently of the first application, by querying the database for such original data, the second application expressing interest in such original data to the first application via a server, receiving any subsequently modified versions of such data having the second format directly from the first application, and storing the subsequently modified versions of such data having the second format in the second cache memory and in the database if the second application sets the persistant storage state.

12 Claims, 88 Drawing figures
Exemplary Claim Number: 8
Number of Drawing Sheets: 45

BRIEF SUMMARY:

1 BACKGROUND OF THE INVENTION

2 The subject matter of the present invention relates to an integrated data communication and data access system, and more particularly, to a data communication system adapted for practicing an event in a first application, creating a data object by the first application during the practice of the event, and communicating between the first application and a cache memory and a database following the practice of the event by independently storing the data object in the cache memory and in the database during a persistant or transient storage state; and to a data access system adapted for independently accessing the database by a second application to retrieve the data object, determining an interest by the second application in subsequently created ones of the data object, expressing the second application's interest in the data objects by transmitting an interest object from the second application to the first application via a server application, and transmitting the subsequently created ones of the data object directly from the first application to the second application.

3 The "Description of the Preferred Embodiment" is divided into two parts: (1) Part 1 (Distributed Framework for Intertask Communication Between Workstation Applications) which is set forth in prior pending application Ser. No. 08/758,833 filed Dec. 4, 1996 and entitled "Distributed Framework for Intertask Communication Between Workstation Applications" and in prior pending provisional application Ser. No. 60/023,945 filed Aug. 19, 1996 and entitled "Distributed Framework for Inter-Process Communication Between Workstation Applications", and (2) Part 2 (Integrated Data Communication and Data Access System including the Application Data Interface) which is set forth in prior pending provisional application Ser. No. 60/023,689 filed Aug. 15, 1996 and entitled "Application Data Interface (ADI)".

4 Computer progams that operate with a network often have multiple programs operating concurrently. It is frequently necessary for information, such as events, to be transferred from one program to another, either within a single workstation or across a network of interconnected computer workstations. An operator at one of the workstations may process information by using different programs operating concurrently in the workstation or across the network of workstations. The operator may also retrieve information by using a multiple number of computer programs executing concurrently in the single workstation or across the network of interconnected workstations. It is therefore important that information be quickly and easily transferred between the multiple number of programs operating in the one or more interconnected workstations.

5 Windowing software technology is applied where it is important for an operator to display and interact with multiple programs executing concurrently in a computer system comprising one or more interconnected workstations. A "window" is defined to be a portion of a display screen, such as a cathode ray tube (CRT). The window covers less than the entirety of the screen. As a result, there may be a multiple number of windows on the screen at one time. Typically, the user moves a cursor around the screen by use of an input device known as a mouse or by use of multiple keys on a keyboard. The cursor can be moved from one window to another on the screen, and, when the cursor is present within a particular window on the screen, the user/operator is placed in communication with the application program which generated that particular window on the screen. As a result, the operator may access a multiple number of different application programs thereby accomplishing multiple tasks without having to load a new program each time a new task must be performed.

6 However, when concurrently accessing a multiple number of different application programs executing in a workstation or across a network of workstations, is is often necessary for a user/operator to transfer information from one windowed program executing in a first workstation to another windowed program executing in either the first workstation or in a second, different workstation connected

to the first workstation across the network. Transferring information between programs operating in a windowing environment is disclosed in this specification; however, such a windowing environment is not necessary in order to practice the invention of this specification as claimed.

7 There are at least three conventional techniques for transferring information between concurrently operating programs in a computer system.

8 The first conventional technique is called "cut and paste". This comprises pointing to and selecting information such as text or data in one window to highlight it and thereby separate it from the remaining information in the window. The user presses a special button or key which moves the selected information to an area of memory specially designated by the operating system and known as the "paste memory" or "clipboard". The user then moves the cursor to another window which is adapted to receive the information. A "paste button" or command is invoked by the user to retrieve the stored information from the designated memory area and place it at the location of the cursor. Note that all steps of this process are carried out by the user.

9 The second conventional technique establishes a programmed connection between two programs, each of which may display information in a window. Both programs must be designed to respond to a predetermined input command that causes information to be shifted from one program to another. This operation may also be entirely under the control of the user and requires a user input in order to function. Each communication path between pairs of programs must be programmed into the code of both programs, which creates an inflexible system. It is also difficult to add new communication paths or to change existing communication paths.

10 The third conventional technique is described in U.S. Pat. No. 5,448,738 to Good et al issued Sep. 5, 1995, and in European Patent Application number 0 380 211 published on Aug. 1, 1990 and entitled "Method for Information Communication between Concurrently Operating Computer Programs" to William E. Good et al (hereinafter called "the Good et al disclosure"), the disclosures of which are incorporated by reference into the specification of this application. In the Good et al disclosure, the user interfaces with application programs through one or more window displays and an input device on a computer workstation. One or more information codes and one or more corresponding application programs are registered with a dispatcher program (otherwise known as a "server"). As a result, a list is formed in the dispatcher program, the list including a plurality of information codes and a corresponding plurality of application program identifiers. Then, when a first application program wants to send event information to another concurrently executing second application program, a template, which includes event information and a corresponding information code, is generated by the first application program and the template is transmitted by the first application program to the dispatcher program. The information code in the template from the first application program is compared with the information code in the list registered with the dispatcher program. If a match between information codes is found, the event information associated with the information code in the template is transmitted to the application program that is associated with the information code in the list registered with the dispatcher program.

11 The Good et al disclosure is similar to other conventional, prior art tools for enabling inter-process communication between application programs, all of which are based on "client-server techniques". Examples of such conventional tools include the "X Window System" and Sun's "Tooltalk". In the Good et al disclosure and the other conventional tools, when using the prior art client-server techniques, all of the data to be communicated between concurrently executing computer program applications must be routed through a server program (the "server program" being similar to the "dispatcher program" in the Good et al disclosure). If many concurrently executing program applications exist in the network, the server or dispatcher may have too many event messages to transmit at any one time. This results in slower throughput

as well as increased network traffic. In addition, when using the prior art client-server technique, the user operator executing a first application program can send only certain preselected event information messages to a second application program. That is, the user can send only a fixed set of predefined event information messages which are allowed by the system network; he cannot define or customize his own event information messages for transmission to the second application program. For example, if a font size was changed in one application, using the conventional client-server technique (where all event messages must be routed through the server), there was no way to communicate the changed font size, changed in one application, to any other application in the network because the "changed font size" was not among the fixed set of predefined event information messages allowed for transmission from one application to another application in the network. In addition, when using the conventional client-server technique, particular event information data must always be communicated from a first computer program application to a server program and from the server program to a second program application. Yet, that server program may not know if any other program applications are interested in receiving that particular event data. As a result, when the server receives the particular event information data from the first program application, (the server must then determine if any other applications are interested in receiving that particular event data. If no other applications are interested in receiving the particular event data, the server program wasted its resources in handling the particular event data because it will never be communicated to any other application.

12 Furthermore, data communicated while using the conventional tools are poorly structured (forming "globs") and provide no mechanism for checking for errors in the data communicated. It is the responsibility of the application programs to interpret the data in the correct manner and handle error conditions. Therefore, when using the conventional tools, the ability of the application programmer to inter-communicate complex data structures between concurrently executing computer program applications is very limited.

13 As noted earlier, the conventional tools do not provide a mechanism which would allow application programmers to selectively extend or customize the type of events and/or data which could be inter-communicated between concurrently executing applications executing in one or more computer workstations. As a result, the absence of a sufficiently high level of programming abstraction in these conventional tools requires application programmers to be concerned with low level issues, such as data formats on various platforms and communication rendevous (such as, a known property name of a known window, as in the prior art "X Window System").

14 Finally, these conventional tools provide no easy way for end-users of applications to control the flow of data and visualize the connectivity between applications.

15 In addition, the conventional tools had a different interface relative to the invention of this application in connection with methods for dealing with events and persistant storage. When data was written into an executing first application, the writer of that data was required to take specific steps, during the introduction of that data, toward the communication that data to another second application. When the data was introduced to the first application, that data was always communicated to the second application via the dispatcher application; and when the receiving second application received that data from the dispatcher application, the receiving second application did not possess a fine "granularity" with regard to the type of data received. That is, the receiving second application could, for example, receive "well data"; however, the receiving second application could not receive a specific subset or type of that well data.

16 Therefore, a "new framework" was needed that would allow fast communication and sharing of complex data, allow strong typing of data structures, and provide a degree of extensibility when using application-defined data types and events.

In addition, that "new framework" should also allow end users of workstation applications to visualize the connectivity network between workstation applications by enabling the end users sitting at those workstations to control the inter-process data communication between computer program applications which are concurrently executing in one or more computer workstation environments.

17 The above referenced "new framework" is disclosed in a prior pending application Ser. No. 08/758,833 filed Dec. 4, 1996 entitled "Distributed Framework for Intertask Communication Between Workstation Applications", to Shamim Ahmed and Serge J. Dacic (hereinafter called the "ITC Application"), the disclosure of which is incorporated by reference into this specification.

18 Although not disclosed in the "ITC Application", the "new framework" utilizes an underlying apparatus called the "Application Data Interface" disclosed in this specification. When the "Application Data Interface" is embodied in the Integrated Data Communication and Data Access System of the present invention, a first application of that System is allowed to independently inquire about the existance of certain newly created data in a database, independently of a second or third application concurrently executing in that System, for the ultimate purpose of possibly expressing interest in and receiving any subsequently created and updated versions of that data which may be generated by the second or third application of that System.

19 However, conventional systems do not allow a first application in the system to inquire about the existance of certain stored data independently of any other applications concurrently executing in the system. For example, if a conventional system includes concurrently executing first and second applications, if the first application is interested in inquiring about the existance of certain data, it must do so by querying the second application. The first application cannot inquire about the existance of that certain data independently of the second application.

20 Accordingly, there is a need for an integrated data communication and data access system, including at least a first and a second concurrently executing application program, that will allow the second application to store certain data in a cache memory and a database when the second application sets persistant or a transient storage state thereby allowing the first application to independently inquire about the existance of such certain data, independently of the second application, by querying the database for such certain data for the ultimate purpose of expressing interest in and receiving any subsequently modified versions of such certain data.

21 SUMMARY OF THE INVENTION

22 Accordingly, it is a primary object of the present invention to provide an integrated data communication and data access system which includes at least a first application and a second application concurrently executing in said system and which is adapted for allowing the second application to store certain data in a second cache memory and a database when the second application sets a persistant or a transient storage state thereby allowing the first application to independently inquire about the existance of such certain data, independently of the second application, by querying the database for such certain data for the ultimate purpose of expressing interest in and receiving any subsequently modified versions of such certain data.

23 It is a further object of the present invention to provide the above referenced integrated data communication and data access system which is additionally adapted for allowing the second application to store the certain data in the second cache memory but not in the database when the second application sets a memory storage state.

24 It is a further object of the present invention to provide the above referenced

integrated data communication and data access system which further includes a first cache memory operatively connected to the first application and a first conversion means operatively connected to the first cache memory and interfacing between a first operator of the first application and the first cache memory for receiving data having a first format from the first operator and converting the data having the first

25 format into data having a second format, the first application storing the data having the second format in the first cache memory and in the database when the first application sets the persistant or transient storage state, the second application independently inquiring about the existance of such data having the second format, independently of the first application, by querying the database for such data having the second format, expressing interest in such data having the second format, receiving any subsequently modified versions of such data having the second format, and storing the subsequently modified versions of such data having the second format in the second cache memory.

26 It is a further object of the present invention to provide the above referenced integrated data communication and data access system which further includes a second conversion means operatively connected to the second cache memory and interfacing between a second operator of the second application and the second cache memory for receiving such data having the second format from the second cache memory and converting such data having the second format into data having a third format, the data having the third format being presented to the second operator of said second application.

27 In accordance with these and other objects of the present invention, an integrated data communication and data access system in accordance with the present invention includes: a first application, a first cache memory operatively connected to the first application, and a first conversion unit operatively connected to the first cache memory and interfacing between a first operator of the first application and the first cache memory; a second application, a second cache memory operatively connected to the second application, and a second conversion unit operatively connected to the second cache memory and interfacing between a second operator of the second application and the second cache memory; a database operatively connected to the first cache memory and the second cache memory; and a server operatively connected to the first application and the second application.

28 In operation, the first operator practices an "event" and the practice of that "event" introduces data having a first format into the first conversion unit. The first conversion unit converts the data having the first format into other data having a second format. The first application receives the data having the second format and stores the data having the second format in the first cache memory and in the database when the first or second application sets a persistant or a transient storage state. On the other hand, the first application stores the data having the second format in the first cache memory but does not store such data in the database when a memory storage state is set. The second application queries the database for the data having the second format and the second conversion unit converts the data having the second format into other data having a third format, the data having the third format being presented to the second operator of the second application. The second operator introduces an interest object into said conversion unit thereby expressing interest in any "subsequently modified versions of the data" generated by the first application, but the interest object does not undergo conversion in the conversion unit. The second application transmits the interest object to the server and the server retransmits the interest object to the first application. The first conversion unit receives the interest object, but the interest object does not undergo conversion in the first conversion unit, and the interest object is presented to the first operator of the first application. The first operator repractices the same "event" and the repractice of that same "event" introduces the "subsequently modified versions of the data" having the first format into the first conversion unit. The conversion

unit converts the "subsequently modified version of the data having the first format" into a "subsequently modified version of the data having the second format", and the Subsequently modified data of the second format is stored in the first cache memory. If the first application sets the persistant storage state, the "subsequently modified version of the data having the second format" will also be stored in the database. However, if the first application sets either the transient or the memory storage state, the "subsequently modified version of the data having the second format" will not be stored in the database.

29 The storage state rules are as follows: an original data set is stored in the cache memory and the database during the persistant or transient storage state, but any subsequently modified data, generated after the original data set is generated, will not be stored in the database during the transient storage state. In addition, both the original and the modified data will not be stored in the database during a memory storage state.

30 When the "subsequently modified version of the data having the asecond format" is stored in the first cache memory associated with the first application, that data will be transmitted directly from the first application to the second application without registering that data with the server. When the second application receives that data, the "subsequently modified version of the data having the second format" will be stored in the second cache memory, and such data having the second format will then be introduced into the second conversion unit. The second conversion unit will convert the "subsequently modified version of the data having the second format" into a "subsequently modified version of the data having a third format". As a result, the "subsequently modified version of the data having the third format" will be presented to the second operator of the second application for his consideration.

31 Further scope of applicability of the present invention will become apparent from the detailed description presented hereinafter. It should be understood, however, that the detailed description and the specific examples, while representing a preferred embodiment of the present invention, are given by way of illustration only, since various changes and modifications within the spirit and scope of the invention will become obvious to one skilled in the art from a reading of the following detailed description.

DRAWING DESCRIPTION:

BRIEF DESCRIPTION OF THE DRAWINGS

A full understanding of the present invention will be obtained from the detailed description of the preferred embodiment presented hereinbelow, and the accompanying drawings, which are given by way of illustration only and are not intended to be limitative of the present invention, and wherein:

FIGS. 1 through 32 are presented in connection with the first part of this specification entitled the "Distributed Framework for Intertask Communication Between Workstation Applications", of which FIGS. 1-32 illustrate the following:

FIG. 1 illustrates a computer workstation having a display which includes one or more window displays representing the execution of one or more client application programs;

FIG. 2 illustrates a plurality of client application programs which display a corresponding plurality of window displays on one or a plurality of workstations similar to that shown in FIG. 1, each of the plurality of client applications being interconnected together by an intertask communication (ITC) apparatus which comprises a server program application and one or more individual client applications;

FIG. 3 illustrates a first workstation executing a first client application and a second workstation executing a second client application and also storing the server application software;

FIG. 4 illustrates a more detailed construction of the first client application (client 1 software) and the second client application (client 2 software) in the first and second workstations of FIG. 3;

FIG. 5 illustrates a more detailed construction of the ITC-Human Interface Code of FIG. 4;

FIGS. 6 through 13b illustrate a detailed construction and the functional operation of the ITC Framework (ITC Core) of FIG. 4;

FIGS. 14 through 25 illustrate a detailed construction of the Operator Interaction Display Software of FIG. 5,

FIG. 14 illustrating the status icons and the broadcast icon,

FIGS. 15a through 15e illustrating the event filter icon with its subwindow display, and

FIGS. 16-25 illustrate the use of these icons on a window display being displayed on a display screen of a workstation;

FIGS. 26 and 27 illustrate a detailed construction of the ITC HI Setup software of FIG. 5;

FIG. 26A illustrates a detailed construction of the "Build List of ITC Events" 80 which is illustrated in FIG. 26;

FIGS. 28 and 29 illustrate a detailed construction of the Send An Event Software of FIG. 5;

FIGS. 30 and 31 illustrate a detailed construction of the Receive An Event Software of FIG. 5;

FIG. 32 illustrates an Intertask Communication (ITC) Sessions Model referenced in the ITC Framework portion of the "Detailed Description of the Preferred Embodiment";

FIGS. 33 through 76 are presented in connection with the second part of this specification entitled the "Integrated Data Communication and Data Access System including the Application Data Interface", of which FIGS. 33-76 illustrate or include the following:

FIGS. 33 through 35 illustrate again the drawings of FIGS. 6, 8A, and 9A,

FIGS. 36 through 44 are presented for reference in conjunction with a general discussion of the concepts associated with the "Integrated Data Communication and Data Access System" of the present invention, and

FIGS. 45 through 76 are presented for reference in conjunction with a detailed discussion of the "Application Data Interface" which is embodied in the "Integrated Data Communication and Data Access System" of the present invention.

DETAILED DESCRIPTION:

1 DESCRIPTION OF THE PREFERRED EMBODIMENT

2 This specification is divided into two parts:

3 (1) a first part (Part 1) that reviews the structure and the functional operation of an invention entitled "Distributed Framework for Intertask Communication Between Workstation Applications" which is set forth in a prior pending application Ser. No. 08/758,833, filed Dec. 4, 1996 (hereinafter called the "ITC Application"), the disclosure of which is incorporated by reference into the specification of this application; the invention of the "ITC Application" describes how direct inter-task communications (ITC) are achieved between concurrently operating computer program applications executing in one or more computer workstations that provide a window display to an operator; and

4 (2) a second part (Part 2) in accordance with the present invention set forth below in this specification entitled "Integrated Data Communication and Data Access System including the Application Data Interface".

5 Part 1--a Distributed Framework for Intertask Communication Between Workstation Applications

6 The following description of the "Distributed Framework for Intertask Communication Between Workstation Applications" (hereinafter called the "ITC Application") is fully disclosed in the prior pending application Ser. No. 08/758,833, filed Dec. 4, 1996, already incorporated herein by reference.

7 Referring to FIG. 1, a computer workstation is illustrated which has a display that includes one or more window displays representing the execution of one or more client application programs.

8 In FIG. 1, a computer workstation 10 is illustrated. The workstation includes a monitor 12, a processor 14, a keyboard 16, and a mouse 18. The monitor 12 includes a cathode ray tube (CRT) that provides a display screen 12a. In FIG. 1, the display screen 12a has a plurality of windows 12b displayed thereon, each window 12b being displayed on the display screen 12a in response to the execution, within the workstation 10, of a separate client application program. As noted below in FIG. 2, each of the plurality of windows 12b provide a different display of wellbore-related data from which an operator can interpret whether or not underground deposits of hydrocarbons are present within an earth formation. An operator sitting at the workstation 10 will use the mouse 18 to select various ones of the windows 12b for viewing and/or manipulation or selection of the data in that window.

9 Referring to FIG. 2, a plurality of client application programs (hereinafter called "client applications") are illustrated which display a corresponding plurality of window displays on one of a plurality of workstations similar to the workstation 10 shown in FIG. 1 and which are interconnected together by an intertask communication (ITC) apparatus.

10 In FIG. 2, a plurality of client applications 20 are interconnected together by an intertask communication (ITC) apparatus 22. Recall that a "client application" is a computer program which is executing in the workstation 10 of FIG. 1 and which is responsible for displaying one of the windows 12b on the display screen 12a of the monitor 12 of the workstation 10 of FIG. 1. The ITC apparatus 22 of FIG. 2 allows each of the client applications 20 to communicate directly with one another. In general, as shown in FIG. 6, the ITC apparatus 22 is a generic term which includes a first client application and a second client application which are interconnected together in the manner shown in FIG. 6. The ITC apparatus 22 of FIG. 2 enables all of the client applications 20 to communicate directly and simultaneously with one another. As a result, events being practiced in one client application 20 can be viewed simultaneously in another client application 20. This functional capability will be discussed in more detail later in this specification.

11 Referring to FIG. 3, a system including a pair of interconnected workstations (each of which are similar to the workstation 10 of FIG. 1) is illustrated.

12 In FIG. 3, a first workstation 24 is interconnected to a second workstation 26. The first workstation 24 includes a processor 24a connected to a system bus 24d, a display 24b (similar to the display screen 12a of FIG. 1) connected to the system bus 24b, a memory 24c connected to the system bus 24d, and a user interface 24e (the mouse 18 and keyboard 16 of FIG. 1) connected to the system bus 24d. The second workstation 26 includes a processor 26a connected to a system bus 26d, a display 26b (similar to the display screen 12a of FIG. 1) connected to the system bus 26b, a memory 26c connected to the system bus 26d and a user interface 26e (the mouse 18 and keyboard 16 of FIG. 1) connected to the system bus 26d. The first workstation 24 is electrically or optically connected to the second workstation 26 via a communication link 28. The memory 24c of the first workstation 24 stores a first client application program called the "client 1 application software" 24c1, and the memory 26c of the second workstation 26 stores a second client application program called the "client 2 application software" 26c1. However, the memory 26c of the second workstation 26 also stores a server software 26c2. The server software 26c2 distributes interest objects between client applications. See the "dispatcher program" which is discussed in the Good et al disclosure (i.e.--in U.S. Pat. No. 5,448,738 entitled "Method for Information Communication between Concurrently Operating Computer Programs"), the disclosure of which has already been incorporated herein by reference. The client 1 application software 24c1, when executed by the processor 24a, generates a visual image on the display 24b which is similar to one of the client applications 20 shown in FIG. 2. Similarly, the client 2 application software 26c1, when executed by the processor 26a, generates a visual image on the display 26b which is similar to one of the other client applications 20 shown in FIG. 2. The function of the system shown in FIG. 3 will become apparent from a reading of the remaining parts of this specification.

13 Referring to FIG. 4, a more detailed construction of the client 1 application software 24c1 and the client 2 application software 26c1 of FIG. 3 is illustrated.

14 In FIG. 4, the client 1 application software 24c1 and the client 2

15 application software 26c1 each include: (1) a first set of software hereinafter known as the "ITC Human Interface Code" 32, and (2) a second set of software hereinafter known as the "ITC Framework (ITC Core) Code" 34. From a functional standpoint, an internal application code invokes the Human Interface Code 32, and the Human Interface Code 32 invokes the Framework Code 34. The ITC Human Interface code 32 and the ITC Framework Code 34 are discussed in greater detail later in this specification and the function of the Human Interface code 32 and the Framework Code 34 will become apparent from a reading of the remaining parts of this specification.

16 Referring to FIG. 5, a more detailed construction of the ITC Human Interface Code 32 of FIG. 4 is illustrated.

17 In FIG. 5, the Human Interface Code 32 of FIG. 4 is comprised of four parts: (1) an Operator Interaction Display Software 32d, (2) an ITC-HI Setup software 32a operatively connected to the Operator Interaction Display software 32d, (3) a Send An Event software 32b operatively connected to the ITC HI Setup software 32a, and (4) a Receive An Event Software 32c operatively connected to the ITC HI Setup software 32a. The ITC Framework (ITC Core) code 34 will be operatively connected to both the Send an Event software 32b and the Receive an Event software 32c. The Operator Interaction Display software 32d will generate displays of icons on the windows 12b of the display screen 12a, and it will also display, on the windows 12b, the "event information" which is requested from other client applications (this will be discussed later in this specification).

18 In operation, referring to FIG. 5, an operator at workstation 10 of FIG. 1, which is executing a particular client application, will view a variety of

icons on a window display 12b on the display screen 12a of the FIG. 1 workstation for that particular client application. The operator will click "on" one or more of the icons thereby allowing an interest object in particular "event information" to be transmitted from the particular client application (e.g.--from the client 1 application 24c1 of FIG. 3) to other client applications (e.g.--client 2 application 26c1 of FIG. 3) via the server 26c2. The operator interaction display software 32d of FIG. 5 will display the window display 12b and the one or more icons in the window display 12b on the display screen 12a of the FIG. 1 workstation. When the operator clicks "on" the one or more icons in the window display 12b, the operator interaction display software 32d will inform the ITC-HI Setup Software 32a, and the ITC-HI Setup Software 32a will drive the Send An Event software 32b. The Send An Event Software 32b will instruct the ITC-Framework Code 34 of the particular client application (e.g.--client 1 application) to send the interest object in the particular event information to the other client applications (e.g.--client 2 application) via the server 26c2. When the other client applications generate the requested event information, the other client applications will send the requested event information directly to the ITC Framework Code 34 of the particular client application without passing through and registering with the server 26c2. When the requested event information is received by the particular client application (e.g.--client 1), the ITC Framework Code 34 of the particular client application will, in turn, inform the Receive An Event Software 32c of the particular client application. The Receive An Event Software 32c of the particular client application will inform the ITC-HI Setup Software 32a of the particular client application that the requested event information has been received from another client application. The ITC HI Setup Software 32a of the particular client application will, in turn, instruct the Operator Interation Display Software 32d of the particular client application to display the requested "event information" on the display screen 12a of the workstation 10 of FIG. 1.

19 Each of these four parts of the Human Interface Code 32 will be discussed later in this specification.

20 Referring to FIGS. 3, and 6 through 13b, a functional operation of the ITC Framework (ITC Core) Code 34 of each client application, such as the client 1 application 24c1 and the client 2 application 26c1, is illustrated.

21 Each client application includes the ITC Framework (ITC Core) Code 34. For example, the client 1 application software 24c1 and the client 2 application software 26c1 of FIG. 3 each include an ITC Framework Code 34. The ITC Framework code 34 associated with any particular client application interacts functionally with the server 26c2 and the ITC Framework code 34 associated with each other client application. For example, the ITC Framework code 34 of the client 1 application 24c1 in the first workstation 24 of FIG. 3 interacts functionally with the server software 26c2 and the ITC Framework code 34 of the client 2 application software 26c1 in the second workstation 26. The Framework code 34 of a first client application 24c1 will transmit interest objects to the server 26c2; it will transmit event information associated with an event "X" to the Framework code 34 of a second client application 26c1; and it will receive event information associated with an event "X" from the Framework code 34 of the second client application 26c1. This functional interaction between the Framework code 34 of one client application 24c1 and the server 26c2 and the Framework code 34 of other client applications 26c1 is discussed in further detail in the following paragraphs with reference to FIGS. 3 and 6 through 13b of the drawings.

22 In FIGS. 3 and 6, referring initially to FIG. 6, a high level schematic for distribution of events and interests is illustrated.

23 In FIG. 3, note the location of the client 1 application software 24c1 in the first workstation 24, and note the location of the client 2 application software 26c1 and the server software 26c2 in the second workstation 26.

24 In FIG. 6, a Client Application 1 24c1 (hereinafter called "Application 1") is interconnected to a Client Application 2 26c1 (hereinafter called "Application 2"), both Application 1 and Application 2 being connected to an ITC server 26c2. The "Client Application 1" 24c1 of FIG. 6 represents the client 1 application software 24c1 of FIG. 3, the "Client Application 2" 26c1 of FIG. 6 representing the client 2 application software 26c1 of FIG. 3, and the ITC server 26c2 of FIG. 6 representing the server software 26c2 of FIG. 3. When the Client Application 1 24c1 of FIG. 6 is executing in the first workstation 24 of FIG. 3, a window display (similar to one of the window displays 12b of FIG. 1) will appear on the display screen of the monitor of the first workstation 24. Similarly, when the Client Application 2 26c1 of FIG. 6 is executing in the second workstation 26 of FIG. 3, another window display (similar to one of the window displays 12b of FIG. 1) will appear on the display screen of the monitor of the second workstation 26. The window displays appearing on both monitors of the first and second workstations 24 and 26 could be any of those shown in FIG. 2.

25 In FIG. 6, assume that the Client Application 1 24c1 ("Application 1") is executing a first client application. In addition, assume that the Client Application 2 26c1 ("Application 2") is executing a second client application and, during the execution of the second client application by Application 2, certain "event information" will be generated by Application 2.

26 The term "event information" and/or the term "event" will be defined in the next three paragraphs by way of the following three examples.

27 For a first example, during the execution of the second client application by Application 2 at workstation 26, the operator sitting at the Workstation 26 may use the mouse 18 of FIG. 1 to place a cursor over one of the windows 12b of FIG. 1 and to thereby select certain information in that window 12b. The "selection" of certain information on that window 12b by Application 2 at workstation 26 may involve either dragging the cursor or deleting information or creating information. The "selection" of that certain information in that window 12b would be an "event" and that event would generate "event information". Application 1 may be interested in receiving that event information.

28 For a second example, Application 1 is being executed in France and it involves an examination of the geology of the earth. Application 2 is being executed in Houston and it involves a petrophysical application that is examining a pressure in a wellbore. There is nothing in common between Application 1 and Application 2 except for one one parameter: the pressure at a certain depth in the earth. The Application 1 may want to examine the pressure v. depth curve being generated in Application 2. Therefore, the pressure v. depth curve in Application 2 and any changes made thereto by an operator at the second workstation 26 would constitute "event information". Application 1 would be interested in receiving that event information.

29 For a third example, called "cursor tracking", Application 1, executing in the first workstation 24 of FIG. 3, is displaying a map having x, y, and z coordinates and an operator at the first workstation 24 can move a cursor across the map which would generate x, y, and z "event information". However, Application 2, executing the second workstation 26 of FIG. 3, is viewing a three-dimensional "cube" representation of an underground reservoir and Application 2 may be interested in receiving the x, y, and z "event information" from Application 1 whenever that event information is generated by Application 1. Therefore, when the operator at the first workstation 24 executing Application 1 moves the cursor across the map, the x, y, and z "event information" is generated from Application 1. If Application 2 previously expressed an interest in receiving that "event information" and when the event information is generated by Application 1, the Application 1 in the first workstation 24 would send that "event information" to Application 2 in the second workstation 26.

30 In FIG. 6, when Application 1 24c1 is executing the first client application, assume that Application 1 is interested in receiving certain "particular event information" from Application 2 26c1 when Application 2 generates that particular event information. In that case, Application 1 24c1 generates an "interest object" signal (which is propagated along line 36 in FIG. 6) from Application 1 24c1 to the ITC server 26c2. The server 26c2 informs Application 2 26c1 of the Application 1 interest in the particular event information by re-propagating the aforementioned "interest object" signal from the server 26c2 to the Application 2 26c1 (along line 38 in FIG. 6). The interest object signal from Application 1 contains an identifier which uniquely identifies Application 1. Therefore, when Application 2 receives the interest object signal (from line 38 in FIG. 6), Application 2 26c1 knows that Application 1 24c1 was interested in receiving the "particular event information" when the particular event information is generated by Application 2. As a result, when Application 2 generates the "particular event information" (i.e.--the operator at workstation 26 selects something by placing the mouse 18 in a window 12b and depressing a key on the mouse), since Application 2 knows that Application 1 is interested in receiving the particular event information, Application 2 will send that particular event information "directly" to Application 1 (via line 40). That is, the particular event information will not be sent from Application 2 26c1 to the server 26c2 (via line 38) and from the server 26c2 to Application 1 24c1 (via the line 36). Because the server 26c2 is not involved in the transfer of the particular event information from Application 2 to Application 1, valuable processing time is saved. As a result, the "Distributed Framework for Intertask Communication Between Workstation Applications" of the present invention is "extensible". That is, for any particular client application program executing in a workstation as represented by one of the windows 12b being displayed in FIG. 1, an application developer can define: (1) the type of events that the particular client application will receive from another concurrently executing client application, and (2) the type of data associated with those events that will be received from the other client application whenever those events are transmitted from the other client applications.

31 In FIG. 7, a flowchart of interest registration and distribution is illustrated. This flowchart discusses some of the concepts discussed above with reference to FIG. 6. In FIG. 7, when Client Application 1 24c1 wants to register an interest in an event (i.e.--Application 1 wants to register an interest in an event because it wants to receive information from one or more other client applications when the other client applications practice or execute the event), a number of process steps are practiced by Application 1 24c1, the server 26c2, and Application 2 26c1:

32 1. In FIG. 7, block 24c1(a), the Client Application 1 24c1 ("Application 1") includes two parts: (1) a first client application ("client application 1"), and (2) an Intertask Communication (ITC) client ("ITC client"). When Application 1 is executing the first client application and if the first client application requires information concerning an event which will hereinafter be called "event X", the first client application will register an interest in event X with the ITC client by sending an "interest object" to the ITC client. The interest object contains an "event token".

33 2. In FIG. 7, block 24c1(b), the ITC client stores certain event tokens. When the ITC client receives the interest object from the first client application, the ITC client will verify the event token present in the interest object by locating a match between the event token in the interest object with an event token catalogued in a database. When a match between event tokens is located, the ITC client will "register an application callback"; that is, the ITC will send an acknowledgement signal back to the first client application.

34 3. In FIG. 7, block 24c1(c), the ITC client will then send an "interest object" signal to the ITC server 26c2.

35 4. In FIG. 7, block 26c2(a), the ITC server will receive the interest object signal from the ITC client and, in response thereto, the ITC server will

register within itself (i.e., the ITC server will store within itself) data or information regarding the interest in event X which the "first client application" of "Application 1" had previously generated. Recall that the first client application of Application 1 previously indicated (by sending the interest object signal to the ITC server) that it wants to receive certain information from the other client applications that is generated when the "event X" is executed or practiced by the other client applications.

36 5. In FIG. 7, block 26c2(b), when the ITC server registers within itself the information regarding the first client application's interest in receiving the event X information from other client applications pursuant to block 26c2(a), the ITC server will broadcast to all such other client applications such first client application's interest. As a result, all the other client applications (i.e.--all the other client application programs being executed in all of the workstations in the network of workstations) will know that the first client application of Application 1 24c1 is interested in receiving certain specific information from the other client applications, the specific information being generated from the other client applications only when the "event X" is executed or practiced by the other client applications.

37 6. In FIG. 7, blocks 24c1(a), 24c1(b), . . . , and 24c1(N), all of the other client applications ("client application 2" 26c1(a), "client application 3" 26c1(b), . . . , and "client application N" 26c1(N)) will register therein (that is, store therein) the first client application's interest in receiving information regarding "event X" whenever any one or all of client application 2, client application 3, . . . , or client application N practice or execute the "event X".

38 In FIGS. 8a-8b, another flowchart of interest registration and distribution (which will discuss concepts similar to the concepts discussed above in connection with the flowcharts of FIGS. 6 and 7) is illustrated. In the flowchart of FIGS. 8a-8b, the "client application 1" 24c1 ("Application 1") is shown to be communicating with the server 26c2 and the "client application 2" 26c1 ("Application 2") as previously illustrated in FIGS. 6 and 7. However, in addition, in FIGS. 8a-8b, two other client applications are illustrated: "client application 3" 42 ("Application 3") and "client application 4" 44 ("Application 4"). In operation, referring to FIGS. 8a-8b, assume for purposes of discussion that Application 1, Application

39 2, Application 3, and Application 4 represent client application programs which are executing in four (4) different workstations similar to the workstation of FIG. 1. Assume further that each of the four applications (Application 1 through Application 4) generate a window display at their respective workstations similar to the window display 12b shown in FIG. 1. Assume further that the Application 1 of FIG. 8a is represented by the "client 1 software" 24c1 in FIG. 3, the Application 2 of FIG. 8a is represented by the "client 2 software" 26c1 in FIG. 3, and the server 26c2 of FIG. 8a is represented by the "server software" 26c2 in FIG. 3. In FIGS. 8a-8b, Application 1 24c1 registers an interest in an ITC event called "event X" (the actual mechanics behind the registration of that interest will be better understood with reference to FIG. 14). An interest object is sent from Application 1 24c1 to the server 26c2 via line 46 in FIG. 8a. When the interest object is received by the server 26c2, the server 26c2 will register, within itself, the interest from Application 1 24c1 in event X. The server 26c2 will then re-distribute the interest object to: "Application 2" 26c1 via line 48, "Application 3" 42 via line 50, and "Application 4" 44 via line 52. When the server 26c2 re-distributes the interest object from Application 1 to the other client applications, Applications 2, 3, and 4, the other client applications (Applications 2, 3, and 4) will register within themselves Application 1's interest in the event X.

40 In FIGS. 9a-9b, a high level schematic of event propagation using peer to peer communication is illustrated. Recall from FIGS. 8a-8bthat Application 1 24c1 transmitted an interest object signal to the server 26c2 and the server 26c2

redistributed that interest object signal to Application 2 26c1. The interest object signal (which identifies Application 1 as being its originator) was registered in Application 2 as originating from Application 1 and it expressed an interest by Application 1 in certain specific information which would be generated by Application 2 when an "event X" is practiced or executed by Application 2. As a result, in FIGS. 9a-9b, since the interest object signal previously sent to Application 2 by the server 26c2 identified Application 1 as being the requestor of such specific information concerning "event X", when Application 2 26c1 practices or executes the "event X", the aforesaid certain specific information concerning the execution or practice of "event X" will be sent directly from "Application 2" 26c1 to "Application 1" 24c1 (that is, the aforesaid certain specific information will not be transmitted from Application 2 to the server 26c2 and from the server 26c2 to Application 1; the server 26c2 has been bypassed).

41 The transmission of the aforementioned certain specific information (concerning the practice by Application 2 of event X) directly from Application 2 to Application 1, without passing through and registering with the server, is an improvement over the Good et al disclosure of the prior art, referenced in the background section of this specification. Recall that, in the Good et al disclosure, all of the data to be communicated between concurrently executing computer program applications must be routed through an intervening server program or dispatcher program.

42 In FIGS. 10a-10b, a high level schematic depicting the multi-casting of event information from Application 2 to other multiple interested client applications is illustrated.

43 Referring briefly to FIGS. 8a-8b, recall that an interest object was sent from Application 1 24c1 to the server 26c2 via line 46 in FIG. 8a. Recall further that, when the interest object was received by the server 26c2, the server 26c2 registered, within itself, the interest from Application 1 24c1 in event X. The server 26c2 then re-distributed the interest object to: "Application 2" 26c1 via line 48, "Application 3" 42 via line 50, and "Application 4" 44 via line 52. However, now that the interest object, in "event X", was re-distributed from the server 26c2 to Applications 2, 3, and 4, if any one or all of Applications 2, 3, and/or 4 practice or execute the "event X", the responsible Application (2, 3, and/or 4) will transmit information concerning the "event X" directly back to the requestor, which is Application 1 24c1, without re-registering with or passing through the server 26c2 (the server 26c2 remains idle).

44 Therefore, in FIGS. 10a-10b, assume that the requestor of event X was both "Application 1" 24c1 and "Application 4" 44 (Application 1 and Application 4 both previously sent an interest object in "event X" to the server 26c2, and the server 26c2 redistributed that interest object in event X to Application 2). Therefore, Application 2 knows that Applications 1 and 4 are interested in receiving information concerning the practice of "event X". As a result, when "Application 2" 26c1 practices or executes the "event X", information concerning the execution of "event X" will be sent directly from Application 2 to both "Application 1" 24c1 and "Application 4" 44 (that information regarding the execution of event X will not re-register with or be routed through the server 26c2 and, as a result, the server 26c2 will be bypassed).

45 In figures 11a-11b, a high level schematic showing "interest revocation" is illustrated. Assume that Client Application 124c1 (Application 1) previously registered an interest in event X with the server 26c2 (by sending an interest object to the server), and then the server 26c2 redistributed that interest object in event X to Client Application 2 26c1 (Application 2), Client Application 3 42 (Application 3), and Client Application 444 (Application 4). Then, assume that Application 1 wants to revoke its interest in "event X". In order to revoke its interest in "event X", the Application 1 will send a "revocation object" to the server 26c2 (via line 46 in FIG. 11), the "revocation object" representing Application 1's indication to other client

applications that is no longer wants to receive any information concerning the practice or execution, by other applications, of the "event X". In response to the receipt of the revocation object, the server 26c2 un-registers, within itself, Application 1's interest in receiving information regarding the execution, by other client applications, of "event X" when event X is practiced by other applications. Then, the server 26c2 re-distributes the revocation object, originating from Application 1, to all the other client applications; that is, in figure 11a, the server 26c2 re-distributes the revocation object to Application 2 (via line 48), Application 3 (via line 50) and Application 4 (via line 52). When the other client applications (Applications 2, 3, and 4) receive the "revocation object", the other client applications (Applications 2, 3, and 4) will un-register Application 1's interest in "event X". As a result, information concerning the practice or execution, by the other Client Applications 2, 3, or 4, of event X, will no longer be sent to Application 1.

46 In FIGS. 12a-12b, a high level schematic of implicit interest revocation for a terminated client is illustrated. Assume that Application 1 24c1 dies or terminates while it has active interests outstanding. Recall that Application 1 has active interests outstanding because Application 1 previously transmitted an interest object in "event X" (and perhaps other events) to the server 26c2 and the server 26c2 previously redistributed that interest object in event X, and other events, to the other client applications, "Application 2" 26c1, "Application 3" 42, and "Application 4" 44. In FIGS. 12a-12b, if Application 1 dies or terminates, the server 26c2 will notice Application 1's termination. When the server 26c2 notices the termination of Application 1 24c1 (the Application 1 program ceases to execute), the server 26c2 will un-register, within itself, all of the outstanding interests which Application 1 previously sent to the server 26c2 (via line 46). Then, the server 26c2 will distribute a "revocation object" corresponding to all of Application 1's outstanding interests in all events to all other client applications, that is, to "Application 2" 26c1, "Application 3" 42, and "Application 4" 44 in FIG. 12. When the other client applications (Applications 2, 3, and 4) receive the revocation object from the server 26c2 associated with all of Application 1's outstanding interests in all events, the other client applications (Applications 2, 3, and 4 in FIG. 12) will un-register all of the interests in all events which "Client Application 1" 24c1 had previously transmitted to Applications 2, 3, and 4 via the server 26c2.

47 In FIGS. 13a-13b, a high level schematic showing the distribution of outstanding interests in a new client application is illustrated. Assume now that a new client application, "Client Application 5" 50 ("Application 5"), in FIG. 13a, begins execution in a workstation in the network of workstations, similar to the workstation 10 of FIG. 1. When the Application 5 program executes, a window would be displayed on the workstation similar to the window displays 12b of FIG. 1. When Application 5 begins to execute, Application 5 will register itself with the server 26c2 in FIG. 13 by sending a "registration signal" (via line 52 in FIG. 13) from the "Application 5" 50 to the server 26c2. In response to the "registration signal", the server 26c2 will register, within itself, the existance of the new client application, "Application 5" 50. Assume now that "Application 2" 26c1, "Application 3" 42, and "Application 4" 44 in FIG. 13 previously sent to the server 26c2 an "interest object" in an event, called "event X", and perhaps other events. In that case, after the server 26c2 registers within itself the existance of the new client "Application 5", the server 26c2 will redistribute the interest objects originating from all the other client applications (Applications 2, 3, and 4) to the new client "Application 5" (via line 54 in FIG. 13a). In the example of FIG. 13a, the server 26c2 will redistribute to Application 5: (1) the Application 2's previously expressed interest in event X and other events, (2) the Application 3's previously expressed interest in event X and other events, and (3) the Application 4's previously expressed interest in event X and other events (hereinafter called "previously expressed interests"). When new client "Application 5" 50 receives the previously expressed interests (i.e.--the interest objects) in event X and other events (which originated from Application's 2, 3, and 4) from the server 26c2, the "Application 5" will

register, within itself, the previously expressed interests. When such previously expressed interests (i.e.--the interest objects) from Applications 2, 3, and 4 are registered within Application 5, the Application 5 will know that Applications 2, 3 and 4 require certain specific information regarding the execution and/or practice of the "event X" and other events. As a result, when the event X or other events are executed by Application 5, the Application 5 will send that certain specific information directly to Applications 2, 3, and 4 (that certain specific information will not pass through and register with the server 26c2 like it did in the Good et al disclosure).

48 Referring to FIGS. 14 through 25, a detailed discussion and a functional operation of the Operator Interaction Display Software 32d of FIG. 5 is illustrated.

49 In the above paragraphs, the functional operation of the ITC Framework 34 in FIG. 4 was discussed in connection with the client 1 application software and the client 2 application software 24c1 and 26c1 stored in the workstations of FIG. 3. Recall that the ITC Framework 34 of the client application 1 24c1 in FIG. 6 sent any requests for event information (associated with "event X") to the server 26c2, whereupon the server 26c2 sent that request for event information to client application 2 26c1. However, when the client application 2 practices or executes the "event X", the event information associated with event X was sent directly from client application 2 to client application 1 (via line 40 in FIG. 6) without passing through or registering with the server 26c2.

50 However, the operator sitting at the second workstation 26 in FIG. 3 can decide how much, if any, of the event information associated with "event X" will be transmitted from the second workstation 26, and the operator sitting at the first workstation 24 in FIG. 3 can decide how much, if any, of the event information associated with "event X" will be received in the first workstation 24.

51 The operators can make that decision and act upon that decision by utilizing certain "icons" which appear within each window display (12b of FIG. 1) on the display screen (12a of FIG. 1) of their particular workstation (10 of FIG. 1). For example, the operator at a workstation can click on a first icon and enable the transmission or reception of all event information to and from his client application, or the operator can click on a second icon and completely disable the transmission or reception of all event information to or from his client application, or the operator can click on a third icon and selectively choose how much of what kind of event information will be transmitted from or received into his client application.

52 The following paragraphs will describe each icon and its function. The icons will appear at the bottom right hand corner of each window display (12b of FIG. 1) on the display screen (12a of FIG. 1) of the workstation 10. There are three main types of icons: the status icon 60, the broadcast icon 62, and the event filter icon 64. There are three types of status icon: the open state icon 60a, the closed state icon 60b, and the locked state icon 60c.

53 In FIG. 14, the status icons 60 and the broadcast icon 62 are illustrated. The status icons 60 include the open state status icon 60a, the closed state status icon 60b, and the locked state status icon 60c. Each of these icons will be discussed below.

54 Open State Status Icon 60a of FIG. 14

55 The open state status icon 60a is accessible to an operator and it will appear on the bottom right hand corner of a window display (similar to window display 12b of FIG. 1). The operator sitting at a workstation (like workstation 24 or 26 in FIG. 3) would locate a window display (12b of FIG. 1) on the display screen (12a of FIG. 1) and click on the open state status icon 60a which

appears at the bottom right hand corner of the window display 12b. When the operator clicks on the open state status icon 60a of a window display 12b for a particular client application, that particular client application is open and it will receive all event information from other client applications; furthermore, that particular client application is open and it will transmit all event information to other client applications. For example, an operator may change a font size (which is an "event" that generates "event information"). If the open state status icon 60a is clicked "on" by the operator for a particular client application program, the font size change event information will be transmitted to all the other interested client applications that requested the font size change event information (via an interest object in the font size change event sent from the other client applications to the particular client application by way of the intervening server).

56 For example, if an operator sitting at the workstation 24 of FIG. 3 clicks on the open state status icon 60a on the window display 12b of FIG. 1 for the client 1 application 24c1 of FIG. 3, the client 1 application 24c1 will receive all requested event information from the client 2 application 26c1 of FIG. 3, and the client 1 application will transmit all requested event information to the client 2 application 26c1.

57 Closed State Status Icon 60b of FIG. 14

58 The closed state status icon 60b is accessible to an operator and it will appear on the bottom right hand corner of a window display (similar to window display 12b of FIG. 1). The operator sitting at a workstation (like workstation 24 or 26 in FIG. 3) can locate a window display (12b of FIG. 1) on the display screen (12a of FIG. 1) and click on the closed state status icon 60b which appears at the bottom right hand corner of the window display. When the operator clicks on the closed state status icon 60b of a window display 12b for a particular client application, that particular client application is closed and it will not receive any event information from other client applications, and that particular client application is closed and it will not transmit any event information to other client applications.

59 For example, if an operator sitting at the workstation 24 of FIG. 3 clicks on the closed state status icon 60b on the window display 12b of FIG. 1 for the client 1 application 24c1 of FIG. 3, the client 1 application 24c1 will not receive any requested event information from the client 2 application 26c1 of FIG. 3, and the client 1 application will not transmit any requested event information to the client 2 application 26c1.

60 Locked State Status Icon 60c

61 The locked state status icon 60c is accessible to both an operator and to the programmer of a particular client application. The locked state status icon 60c will appear at the bottom right hand corner of a window display (12b of FIG. 1) of the particular client application. In some cases, the operator at a workstation may click "on" the open state status icon 60a in a window display 12b for the particular client application. However, the application programmer may have previously decided that, for the aforementioned particular client application, absolutely no event information can be transmitted from or received in that particular client application. As a result, the application programmer, that programmed the particular client application, may have required (internally within the particular client application code) that the particular client application be closed (as if the closed state status icon 60b were clicked "on"). In that event, the locked state status icon 60c will appear to click "on", by itself, in response to the requirement to close the particular client application, which requirement would be placed inside the particular client application code. An unstable state could cause the locked state status icon 60c to automatically click "on".

62 However, the operator could also click "on" the locked state status icon 60c. When the locked state status icon 60c is clicked "on,", this is equivalent to clicking "on" the closed state status icon 60b. That is, when the locked state status icon 60c is clicked "on", event information will not be received by a particular client application from other client applications (via line 40 in FIG. 6), event information will not be sent from the particular client application to other client applications, and an interest object associated with a particular event will not be send by the particular client application to the server 26c2 (via line 36 in FIG. 6) for further transmission to other client applications (via line 38 in FIG. 6).

63 Broadcast Icon 62

64 The broadcast icon 62 will appear at the bottom right hand corner of a particular window display (12b of FIG. 1) which is generated by a particular client application program, such as client 1 or client 2 of FIG. 3, executing within a workstation (10 of FIG. 1). Assume that, for that particular client application program, the closed state status icon 60b has been clicked "on" for a period of time. A plurality of newly created events (such as, changes to font size, changes to color, or dashed lines) which were generated during that period of time will not be transmitted by the particular client application to other interested client applications executing in the subject workstation or other workstations in the network of workstations. However, when the broadcast icon 62 is clicked "on" within the window display (12b) by an operator sitting at the workstation (10 of FIG. 1) using the mouse (18 of FIG. 1), all of the plurality of newly created events in the particular client application (12b), which were generated by an operator at the workstation 10 during the aforementioned period of time when the closed state status icon 60b was clicked "on", will be transmitted simultaneously from the particular client application to all other "interested client applications" executing in the network of workstations (the words "interested client applications" indicating that "interest objects" in the newly created events were previously transmitted from the other client applications to the server 26c2 and from the server 26c2 to the particular client application).

65 In FIGS. 15a through 15e, the Event Filter icon 64 is illustrated. The event filter icon 64 will be discussed in the following paragraph.

66 Event Filter Icon 64

67 In FIG. 5, recall that each client application, such as the client 1 application 24c1 and the client 2 application 26c1 of FIG. 3, include an ITC-HI Setup software 32a (FIG. 5). The ITC-Setup software 32a includes a coded and stored "list of events" and a "list of functions" corresponding, respectively, to the "list of events" (this list of events and corresponding list of functions will be discussed in greater detail in this specification with reference to FIG. 26 of the drawings).

68 Therefore, when a particular client application (such as the client 1 or client 2 application of FIG. 3) sends an interest object in an "event X" to another client application via the server 26c2 for the purpose of receiving event information from the other client applications regarding the practice of that event X, the "event X" must, of necessity, be one of the events in the "list of events" (of FIG. 26) coded within the ITC Setup Software 32a of FIG. 5 for that particular client application.

69 Similarly, if a particular client application intends to send "event information" to other client applications that is associated with the practice by the particular client application of a "particular event", that "particular event" must be one of the events stored in the "list of events" (of FIG. 26) coded within the ITC Setup Software 32a of FIG. 5 for that particular client application.

70 Consequently, since each particular client application is said to be "interested" in receiving a plurality of "event information" associated with a plurality of events from other client applications, the plurality of events are stored in the "list of events" coded within the ITC Setup software 32a (see FIG. 26 for the "list of events") for said each particular client application. Similarly, since each particular client application will send "event information" associated with a plurality of events to other interested client applications, those plurality of events are stored in the "list of events" coded within the ITC Setup software 32a.

71 However, by using the "Event Filter" icon 64 of FIG. 15, an operator or user, monitoring the particular client application on a window display (12b of FIG. 1) at his workstation (10 of FIG. 1), can selectively decide how many of the plurality of events in the list of events coded within the ITC Setup software 32a he will transmit to other client applications via the server 26c2 and how many of the plurality of events in the list of events coded within the ITC Setup software 32a he will receive from the other client applications via the server 26c2.

72 In FIGS. 15a through 15e, the event filter icon 64 of FIGS. 15a and 15b will be located at the bottom right hand corner of each window display (12b) on the display screen (12a) of a workstation (10). As shown in FIG. 15b, when the operator at the workstation (10) uses the mouse 18 to click "on" the event filter 64 which appear on a window display 12b, a subwindow display 64a shown in FIGS. 15c will be presented to the operator on the display screen (12a).

73 In FIG. 15c, the subwindow display 64a (which appears on the display screen 12a of the workstation 10 when the event filter icon 64 in a window display 12b for a particular client application is clicked "on" by an operator) includes three columns: (1) the send column 64a1, (2) the receive column 64a2, and (3) the message or event column 64a3. A plurality of messages or events 64a3A are printed under the message column 64a3. These plurality of messages or events 64a3A represent a plurality of events for which: (1) a corresponding plurality of event information could be received from other client applications, and (2) a corresponding plurality of event information could be sent or transmitted to other client applications. A plurality of send boxes 64a1A appear under the send column 64a1, and a plurality of receive boxes 64a2A appear under the receive column 64a2. For each of the plurality of messages 64a3A, there is one send box 64a1A and one receive box 64a2A. An operator would use the mouse 18 of FIG. 1 to click within a send box and/or a receive box for each of the plurality of events 64a3A.

74 If the operator clicked within a send box 64a1A for a particular message or event (one of events 64a3A in FIG. 15c) for a particular client application, "event information" associated with that particular event will, in fact, be sent from the particular client application to the other client applications that registered an interest in the particular event with the particular client application; however, if the operator did not click within the send box 64a1A for that particular event 64a3A for that particular client application, "event information" associated with that particular event 64a3A will not be sent from the particular client application to the other client applications that registered an interest in the particular event with the particular client application.

75 In addition, If the operator clicked within a receive box 64a2A for a particular message or event 64a3A for a particular client application, "event information" associated with that particular event 64a3A will, in fact, be received from other client applications in response to the registry by the particular client application with the other client applications in that particular event; however, if the operator did not click within the receive box 64a2A for that particular event 64a3A for that particular client application, "event information" associated with that particular event 64a3A will not be received from the other client applications in response to the registry by the particular client application with the other client applications in that

particular event.

76 In FIGS. 15d and 15e, consider the examples of the use of the event filter 64 and the subsequent use of the subwindow display 64a for that event filter 64 illustrated in FIGS. 15d and 15e.

77 In the example of FIG. 15d, four events appear under the events column 64a3 of the subwindow display 64a of the event filter 64 (appearing in a window display 12b on the display screen 12a of a workstation 10 for a particular client application): (1) change color, (2) change thickness, (3) change shape, and (4) cursor tracking. Note, for each of these events, whether the send boxes 64a1A and/or the receive boxes 64a2A are clicked "on" (by placing a black mark in the box).

78 In FIG. 15d, taking each event in order, for the "change color" event of FIG. 15d, the send box 1A1 is not clicked, and the receive box 2A1 is not clicked. As a result, for the "change color" event for the particular client application, event information for the "change color" event will not be transmitted to other client applications, and event information for the "change color" event will not be received from other client applications.

79 In FIG. 15d, for the "change thickness" event, the send box 1A2 is clicked, but the receive box 2A2 is not clicked. As a result, for the "change thickness" event for the particular client application, event information for the "change thickness" event will be transmitted to other client applications, but event information for the "change thickness" event will not be received from other client applications.

80 In FIG. 15d, for the "change shape" event, the send box 1A3 is not clicked, but the receive box 2A3 is clicked. As a result, for the "change shape" event for the particular client application, event information for the "change shape" event will be not transmitted to other client applications, but event information for the "change shape" event will be received from other client applications.

81 In FIG. 15d, for the "cursor tracking" event, the send box 1A4 is clicked, and the receive box 2A4 is clicked. As a result, for the "cursor tracking" event for the particular client application, event information for the "cursor tracking" event will be transmitted to other client applications, and event information for the "cursor tracking" event will be received from other client applications.

82 However, in the example of FIG. 15e, the same four events appear under the events column 64a3 of the subwindow display 64a of the event filter 64 (appearing in a window display 12b on the display screen 12a of a workstation 10 for a particular client application): (1) change color, (2) change thickness, (3) change shape, and (4) cursor tracking. The subwindow display 64a in FIG. 15e further includes an "all" box 64a4 under the "send" column 64a1 and another "all" box 64a5 under the "receive" column 64a2. When an "all" box is clicked "on", each of the individual (send or receive) boxes above the "all" box will be clicked "on"; however, if the "all" box is not clicked "on", each of the individual boxes above the "all" box must be individually clicked "on" or "off". For example, note that the "all" box 64a5 under the receive column 64a2 is clicked "on", but the "all" box 64a4 under the send column 64a1 is not clicked "on". Since the "all" box 64a5 under the receive column 64a2 is clicked "on", all of the receive boxes 2A1, 2A2, 2A3, and 2A4 in the subwindow display 64ain FIG. 15e are clicked "on". However, since the "all" box 64a4 under the "send" column 64a1 is not clicked "on", each of the individual boxes 1A1, 1A2, 1A3, and 1A4 must be individually clicked as either "on" or "off". As a result, in FIG. 15e, the "change color" event will not be sent from the particular client application to other client applications but it will be received by the particular client application from other client applications. In addition, the "change thickness" event will be sent from the particular client application to

other client applications and it will be received by the particular client application from the other client applications. The "change shape" event will not be sent by the particular client application to other client applications, but it will be received by the particular client application from other client applications. The "cursor tracking" event will be sent by the particular client application to the other client applications and it will be received by the particular client application from the other client applications.

83 The functional operation of the event filter 64 and its subwindow 64a will be set forth again below in connection with a discussion of FIG. 26 and the functional operation of the present invention.

84 In FIGS. 16 through 23, examples of the use of all the icons of FIGS. 14 and 15a, including the open state icon 60a, the closed state icon 60b, the locked state icon 60c, the broadcast icon 62, and the event filter icon 64, are illustrated.

85 In FIG. 16, a window display, which could be one of the window displays 12b of FIG. 1, has a group of icons in the bottom right hand corner of the window display, the icons including a closed state status icon 60b, a broadcast icon 62, and an event filter 64.

86 In FIG. 17, another window display 12b has a closed state status icon 60b and a broadcast icon 62 in the bottom right hand corner of the window display 12b.

87 In FIG. 18, another window display 12b has an open state status icon 60a and a broadcast icon 62 in the bottom right hand corner of the window display 12b.

88 In FIG. 19, another window display 12b has a locked state status icon 60c and a broadcast icon 62 in the bottom right hand corner of the window display 12b.

89 In FIGS. 20 through 23, referring initially to FIG. 20, an operator will view a master window 12b1 on the display screen 12a of FIG. 1 and, by using the mouse 18 of FIG. 1, the operator can subsequently obtain a number of sub-windows shown in FIGS. 21, 22, and 23. For example, the master window 12b1 of FIG. 20 includes an open state status icon 60a and a broadcast icon 62 in the bottom right hand corner of the window. However, the master window 12b1 of FIG. 20 also includes a box 70. If the operator uses the mouse 18 to click on the box 70 in the master window 12b1 of FIG. 20, in addition to the master window 12b1, a first sub-window 12b2 shown in FIG. 21 will be presented to the operator on the display screen 12a of the workstation 10 of FIG. 1. The first sub-window 12b2 includes an open state status icon 60a and a broadcast icon 62 in the bottom right hand corner of the first sub-window 12b2. The first sub-window 12b2 includes a second box 72 and a third box 74. If the operator uses the mouse 18 to click on the second box 72 in the first sub-window 12b2 of FIG. 21, a second sub-window 12b3 shown in FIG. 22 will be presented to the operator on the display screen 12a of the workstation 10 of FIG. 1. The second sub-window 12b3 includes a locked state status icon 60c and a broadcast icon 62 in the bottom right hand corner of the second sub-window 12b3. If the operator uses the mouse 18 to click on the third box 74 in the first sub-window 12b2 of FIG. 21, a third sub-window 12b4 shown in FIG. 23 will be presented to the operator on the display screen 12a of the workstation 10 of FIG. 1. The third sub-window 12b4 includes a closed state status icon 60b and a broadcast icon 62 in the bottom right hand corner of the third sub-window 12b4.

90 The above paragraphs have discussed the structure and function of the status icons which includes the open state status 60a, the closed state

91 status icon 60b, and the locked state status icon 60c, in addition to the broadcast icon 62 and the event filter icon 64. Each of these icons would appear on the bottom right hand corner of a window 12b of FIG. 1. However, there is one additional icon to be disclosed, called the "Raised Application Manager Event" icon, that would appear on the bottom left hand corner of the

window display 12b of FIG. 1 (not the right hand corner where the other icons discussed above appear). The "Raised Application Manager Event" icon is discussed in detail, as follows.

92 Raised Application Manager Event Icon 76

93 In FIGS. 1, 24 and 25, each of the windows 12b of FIG. 1 include a "Raised Application Manager Event" icon 76 (of FIG. 24) which is located in the bottom left hand corner of the window 12b. For example, one of the windows 12b of FIG. 1 could include the window 12b5 of FIG. 24.

94 In FIG. 24, the window 12b5 includes the Raised Application Manager Event icon 76 in the bottom left hand corner of the window 12b5. Assume now that a multitude of windows 12b are being presented to the operator on the display screen 12a of the workstation 10 of FIG. 1. Assume further that the operator wants to access a particular client application from the workstation 10; however, the multitude of windows 12b on the display screen 12a is obscuring the display screen 12a and, as a result, it is very difficult for the operator to access the particular client application.

95 In FIGS. 1 and 24, order to access the particular client application, the operator will find the "Raised Application Manager Event" icon 76 in the bottom left hand corner of any window 12b on the display screen 12a of FIG. 1 One of the windows 12b of FIG. 1 could include the window 12b5 of FIG. 24. The window 12b5 of FIG. 24 includes the "Raised Application Manager Event" icon 76 in the bottom left hand corner and a locked state status icon 60c and a broadcast icon 62 in the bottom right hand corner of the window 12b5. Using the mouse 18, the operator clicks "on" the Raised Application Manager Event icon 76 of FIG. 24. In response, a main launch window 12b6, illustrated in FIG. 25, is displayed on the display screen 12a of the workstation 10 of FIG. 1.

96 In FIGS. 2 and 25, the main launch window 12b6 of FIG. 25 includes a plurality of different client application icons 78 representing a respective plurality of different client application programs. The workstation 10 of FIG. 1 will execute any particular one of the different client application programs if and when the operator at the workstation 10 of FIG. 1 uses the mouse 18 to click "on" the client application icon 78 of FIG. 25 which corresponds to that particular client application program. See, for example, the different client applications 20 shown in FIG. 2. Each of the plurality of different client applications 78 shown in FIG. 25 could represent one of the plurality of client applications 20 shown in FIG. 2.

97 Referring to FIGS. 26, 26A, and 27, a detailed construction and a functional operation of the ITC HI Setup software 32aof FIG. 5 is illustrated.

98 In FIG. 26, the ITC HI Setup Software 32a of FIG. 5 includes (but is not limited to) the following blocks of code:

99 (1) "Build List of ITC Events" 80,

100 (2) "Function 1 to call on reception of Event 1" 82,

101 (3) "Function 2 to call on reception of Event 2" 84,

102 (4) "Call to itc.sub.-- hi.sub.-- Filter And Session" 86,

103 (5) "Function to call on Broadcast" 88, and

104 (6) "Call to itc.sub.-- hi.sub.-- Delete" 90.

105 The "Build List of ITC Events" 80 block of FIG. 26 is illustrated in greater detail in FIG. 26A.

106 Each of these blocks of code is discussed below.

107 Build List of ITC Events 80

108 Function 1 to call on reception of Event 1 82

109 Function 2 to call on reception of Event 2 84

110 In FIGS. 3, 4, 5, and 6 recall from FIGS. 3 and 4 that the client 1 application 24c1 was stored in the memory 24c of the first workstation 24 of FIG. 3 and the client 2 application 26c1 was stored in the memory 26c of the second workstation 26 of FIG. 3. The client 1 application 24c1 and the client 2 application 26c1 each include: (1) the ITC Human Interface Code 32, and (2) the ITC Framework (ITC-Core) Code 34 of FIG. 4. The ITC Framework (ITC-Core) Code 34 was discussed above with reference to FIGS. 6 through 13. The ITC Human Interface Code 32 of FIG. 4 includes four parts: the ITC HI Setup software 32a of FIG. 5, the Send An Event software 32b of FIG. 5, the Receive An Event software 32c of FIG. 5, and the Operator Interaction Display Software 32d of FIG. 5.

111 When the client 1 application 24c1 wants to receive "event information" regarding "event X" from the client 2 application 26c1, the client 1 application 24c1 will send an "interest object" in event X to the server 26c2 in FIGS. 3 and 6. In response to the receipt by the server 26c2 of the "interest object" in event X from client 1, the server 26c2 will: (1) register within itself the client 1's interest in event X, and (2) redistribute that interest object in event X from the server 26c2 to the client 2 application 26c1. When the client 2 application 26c1 practices or executes the event X, the event information associated with the practice of event X in client 2 will be sent directly from client 2 to client 1 (via line 40 in FIG. 6) without passing through and registering with the server 26c2.

112 Similarly, the client 2 application 26c1 of FIG. 3 will send an interest object in event X to the server 26c2, and the server 26c2 will: (1) register within itself client 2's interest in the event information associated with event X, and (2) send the interest object in event X to the client 1 application 24c1. When the client 1 application 24c1 practices or executes the event X, the event information associated with the event X will be sent directly by client 1 24c1 to client 2 26c1 (via line 40 in FIG. 6) without passing through and registering with the server 26c2.

113 Therefore, each client application program, including the client 1 application 24c1 and the client 2 application 26c1 of FIG. 3, must record and store within itself the identity of all of the specific events (such as "event X"), as well as their interest objects and their functions, in which that particular client application is interested.

114 In FIG. 26, each particular client application program, including the client 1 application 24c1 and the client 2 application 26c1 of FIG. 3 which generated two of the window displays 12b of FIG. 1, stores within itself a "list of ITC events" 80, a "list of functions" 82, 84 corresponding, respectively, to the "list of ITC events" 80, and a "list of interest objects" corresponding, respectively, to the "list of functions" and the "list of ITC events" 80.

115 As a result, in FIG. 26, block 80 stores a list of events called "Build a list of ITC Events" wherein a plurality of events (i.e. , event 1, event 2, event 3, . . . , event N) are stored. Blocks 80, 84 will store a plurality of functions (i.e. , function 1, function 2, function 3, . . . , function N), which correspond, respectively, to the plurality of events of block 80, the plurality of functions being retrieved from memory and executed in response to the reception (by the ITC Framework 34 of FIG. 5 of a particular client application) of a respective plurality of events transmitted to the particular

client application from the other client applications.

116 For example, a first function in FIG. 26 called "Function 1 to call on reception of Event 1" 82 represents the function associated with "event 1" in the block "Build a list of ITC Events" 80. A second function in FIG. 26 called "Function 2 to call on reception of Event 2" 84 represents the function associated with "event 2" in the block "Build a list of ITC Events" 80.

117 In addition, in FIG. 26A, the block 80 of FIG. 26 will also store a plurality of "interest objects" corresponding, respectively, to the plurality of "events". For example, in FIG. 26A, the block 80 of FIG. 26, which is called "Build List of ITC Events", will have at least two columns of stored information: (1) a first column 80a storing a plurality of events 80a, such as Event 1, Event 2, Event 3, . . . , and Event N; and (2) a second column 80b storing a plurality of interest objects 80b which correspond, respectively, to the plurality of events 80a, such as "Interest Object 1" associated with "Event 1", "Interest Object 2" associated with "Event 2", "Interest Object 3" associated with "Event 3", . . . , and "Interest Object N" associated with "Event N".

118 Therefore, when the particular client application program begins to execute, since the particular client application stored within itself a "list of ITC events" 80, the particular client application will send the interest object, associated with each of the events in the stored "list of ITC events" 80, from the particular client application to the other client applications via the server 26c2 as shown in FIG. 6.

119 In addition, if the a particular set of interest objects corresponding to a particular set of events are sent by the particular client application 24c1 to the other client applications 26c1 via the server 26c2, if the "Build a list of events" 80 in the other client applications 26c1 include the aforesaid particular set of events, and when the other client applications 26c1 practice or execute the particular set of events, the other client applications 26c1 will send a particular set of event information associated with the particular set of events directly to the particular client application (via line 40 in FIG. 6) without registering that event information with the server 26c2 and the particular client application will receive that particular set of event information.

120 In addition, when other client applications send an interest object in an event X to the particular client application via the server 26c2, since the particular client application stores within itself a "list of ITC events" 80 and a corresponding list of "interest objects" associated with the list of ITC events 80, the received interest object from the other client application is compared by the particular client application with the "list of interest objects" 80 of FIG. 26A stored in the particular client application, and, if a match if found, the event which corresponds to the matched interest object (i.e. , "event X") will be transmitted from the particular client application directly to the other client applications (directly via line 40 in FIG. 6).

121 In addition, for each particular client application wherein a "list of ITC events" 80 is stored, a corresponding "list of functions" 82, 84 must be stored corresponding, respectively, to the stored "list of ITC events" 80. As a result, when another client application practices or executes the requested "event X", the event information associated with that event X will be sent directly from that other client application to the particular client application (via line 40 in FIG. 6) without passing through and registering with the server. When the event information associated with event X is received by the particular client application, the received event information will be compared by the particular client application with the "list of ITC events" 80 and their corresponding "list of functions" 82, 84 stored in the particular client application. When a match is found, by the particular client application, between the event information received from the other client application and "one particular event" in the "list of ITC events" 80, the

"particular function" 82, 84 which is associated with that "one particular event" will be automatically recalled from memory 24c, 26c, the "particular function" 82, 84 being executed by the processor 24a or 26a of FIG. 3 in the workstation 10 which is executing the particular client application. The Operator Interation Display Software 32d of FIG. 5 will ensure that the "particular function" (i.e, the received event information, such as depth change or color change or line thickness change) will be displayed on the display screen 12a of the workstation 10 of FIG. 1.

122 Call to itc hi Filter And Session 86

123 In FIG. 26, the "Call to itc hi Filter And Session" 86 is the portion of the ITC-HI Setup software 32a of FIG. 5 which performs the "human interface" function.

124 In FIG. 5, note the intermediate location of the ITC-HI Setup" 32a between the "Operator Interaction Display software" 32d, which displays the icons and the event information, and the "Send An Event" 32b and the "Receive An Event" 32c software which sends event information to and receives event information from other client applications.

125 In FIG. 26, since the "Call to itc Filter And Session" 86 is an integral part of the ITC-HI Setup software 32a, it is evident from the intermediate location of the ITC-HI Setup 32a software in FIG. 5 (between the "Operator Interaction Display software" 32d and the "Send An Event" 32b and the "Receive An Event" 32c software) that the "Call to itc hi Filter And Session" 86 portion of the ITC-HI Setup Software 32a in FIG. 26 will function as a coordinator located between two ends for receiving information from one end and, in response thereto, for instructing the other end.

126 For example, in FIGS. 5 and 26, the "Call to itc hi Filter And Session" 86 will receive event information from the "Receive An Event" software 32c (where the event information originated from another client application and is associated with an event X) and, in response thereto, will drive the "Operator Interaction Display" software 32d for displaying that event information associated with the event X on the display screen 12a of FIG. 1 for a particular client application.

127 In addition, in FIGS. 5 and 26, the "Call to itc hi Filter And Session" 86 will receive event information (e.g.--changed parameters, color, thickness, font size) from the "Operator Interaction Display" software 32d of a particular client application and, in response thereto, will drive the "Send An Event" software 32b which will further instruct the ITC Framework 34 to send the aforementioned event information to other interested client applications.

128 Therefore, the "Call to itc hi Filter And Session" 86 will make all the connections; that is: it will send all interest objects from the "Build list of ITC Events" 80 of a particular client application to the server 26c2 for further transmission to other client applications; it will associate event information received from other client applications with a particular function 82, 84 of FIG. 26 in a particular client application for executing that particular function in the particular client application; it will cause the Operator Interaction Display Software 32d to build the various icons (status icons 60, broadcast icon 62, event filter icon 64) for display in a particular client application on the display screen 12a; and it will notify the ITC Framework (ITC Core) 34 of FIG. 5 of a particular client application which will, in turn, notify the server 26c2 that the particular client application is interested in the plurality of events that are listed in its "Build list of ITC Events" 80 of FIG. 26.

129 Function to call on Broadcast 88

130 In FIG. 26, the "Function to call on Broadcast" 88 part of the ITC-HI Setup

Software 32a of FIG. 5 is associated with the "Call to itc.sub.-- hi.sub.-- Filter And Session" 86. In order to explain the function of the "Function to call on Broadcast" 88, it is necessary to recall the function of the "Broadcast" Icon 62 of FIG. 14.

131 When the broadcast icon 62 for a particular client application 12b is clicked "on" by an operator at workstation 10 using the mouse 18, in most cases, all of a plurality of newly created events in the particular client application, which were generated by an operator at the workstation 10 during a period of time after the closed state status icon 60b was clicked "on", will be simultaneously transmitted from the particular client application to all other "interested client applications" executing in the network of workstations.

132 For example, for a particular client application where particular events consisting of color events, font size events, and line thickness events can be transmitted to and received from other client applications, when the operator of the particular client application clicks "on" the broadcast icon 62 after a period of time elapses following the clicking "on" of the closed state status icon 60b, in most cases, event information associated with all of the particular events will be transmitted to the

133 other client applications.

134 However, the "Function to call on Broadcast" 88 will allow the particular client application developer to decide whether or not event information associated with "all" of the particular events will be transmitted to the other client applications when the broadcast icon 62 is clicked "on" by the operator. More particularly, using the "Function to call on Broadcast" 88, event information associated with "some" of the particular events (which were newly created in the particular client application after the closed state status icon was clicked "on" by the operator) will be transmitted to the other client applications when the broadcast icon 62 is clicked "on" by the operator.

135 In FIGS. 5 and 26, when the operator clicks "on" the broadcast icon 62 for a particular client application after a period of time elapsed following the clicking "on" of the closed state status icon 60b, the "Operator Interaction Display" software 32d in FIG. 5 for that particular client application will notify the ITC-HI Setup Software 32a in FIG. 5 that the operator clicked "on" the broadcast icon 62. In response, the "Call to itc.sub.-- hi.sub.-- Filter And Session" 86 portion of the ITC-HI Setup Software 32a in FIG. 26 will refer to and call up the "Function to call on Broadcast" 88 part of the ITC-HI Setup Software 32a.

136 Assume that a "plurality of newly created events" were practiced and executed by the particular client application between the time when the closed state status icon 60b was clicked "on" and the time when the broadcast icon 62 was clicked "on" by the operator executing the particular client application.

137 The "Function to call on Broadcast" 88 will determine whether "all" or "some" of the "plurality of newly created events" will be transmitted to the other client applications in response to the clicking "on" of the broadcast icon 62 by the operator executing the particular client application. The "Function to call on Broadcast" 88 will determine how many events of the "plurality of newly created events" (i.e.--some, all, or none) will be transmitted to the other client applications.

138 In FIGS. 5 and 26, using the above example, for a particular client application where particular events consisting of color events, font size events, and line thickness events can be transmitted to and received from other client applications, when the operator of the particular client application clicks "on" the broadcast icon 62 after a period of time elapses following the clicking "on" of the closed state status icon 60b and when the Operator Interaction Display software 32d of FIG. 5 notifies the ITC-HI Setup Software

32a that the broadcast icon 62 has been clicked "on", the "Call to Itc.sub.-- hi.sub.-- Filter And Session" 86 will call up and retrieve the "Function to call on Broadcast" 88 part of the ITC-HI Setup Software 32a, and, in response thereto, the "Function to call on Broadcast" 88 can require that event information associated with only some of the events, such as only the color events for example, will be transmitted to the other client applications.

139 Call to itc hi Delete 90

140 In FIGS. 5 and 26, when the particular client application ceases to execute (i.e., the operator at the workstation 10 terminates a window display 12b representing the particular client application), the operator interaction display software 32d of FIG. 5 will notify the ITC-HI Setup software 32a. In response thereto, the "Call to itc.sub.-- hi.sub.-- Delete" 90 portion of the ITC-HI Setup software 32awill notify ITC Framework 34 and the ITC Framework 34 will notify the server 26c2 that the particular client application has terminated. In response, the server 26c2 will un-register any and all interest objects stored therein which are associated with the particular client application, and then the server 26c2 will notify all other client applications. In response, the other client applications will un-register the particular client application's interests in certain previously registered events. As a result, the other client applications will not send any event information corresponding to the previously registered events to the particular client application.

141 In FIG. 27, the actual program code which corresponds to the ITC-HI Setup software 32a of FIGS. 5 and 26 is illustrated.

142 Referring to FIGS. 28 and 29, a detailed construction and a functional operation of the Send An Event Software 32b of FIG. 5 is illustrated.

143 In FIG. 28, the Send An Event software 32b of FIG. 5 includes two blocks of code: (1) Get Data Structure to Send 92, and (2) Call to itc.sub.-- hi.sub.-- Transmit Event 94. Each of these blocks of code will be discussed individually.

144 Get Data Structure to Send 92

145 In FIG. 28, the "Get Data Structure to Send" 92 block of code responds to three different types of input data: (1) input data originating from a user interaction, (2) input data originating from a Database, and (3) input data originating from an I/O Stream.

146 In FIGS. 5 and 28, the Operator Interaction Display software 32d responds to any changes which are made to a particular client application by an operator at workstation 10 of FIG. 1 by generating the "user interaction" type of input data which is ultimately input to the "Get Data Structure to Send" 92 block of code associated with the Send An Event software 32b. For example, when the operator at workstation 10 of FIG. 1 is working with a particular client application as represented by one of the window displays 12b of FIG. 1, the operator may change the color, or the font size, or he may make some other change to the particular client application. If those changes are in the list of events in the "build list of events" 80 of FIG. 26 and when another client application has requested event information associated with those changes, the Operator Interaction Display software 32d of FIG. 5 will respond to the changes made by the operator to the particular client application by notifying the ITC-HI Setup software 32a.

147 The "Call to itc.sub.-- hi.sub.-- Filter and Session" 86 portion of the ITC-HI Setup software 32a will provide the following information to the "Get Data Structure to Send" 92 portion of the "Send An Event software" 32b of FIG. 28: (1) the name of the event which is associated with the aforementioned changes which were made by the operator to the particular client application, and (2) the data or event information which is associated with the aforementioned named

event.

148 However, there are two other origins of the information (name of the event, and data or event information associated with the named event) which is provided by the ITC-HI Setup software 32a to the "Get Data Structure to Send" 92 portion of the "Send An Event software" 32b of FIG. 28: (1) input data originating from a Database, and (2) input data originating from an I/O Stream.

149 Call to itc hi Transmit Event 94

150 In FIG. 28, when the "Get Data Structure to Send" 92 portion of the "Send An Event software" 32b of FIG. 28 receives (1) the name of the event which is associated with the changes which were made by the operator to the particular client application, and (2) the data or event information which is associated with the aforementioned named event, a call is made to the "itc hi Transmit Event" 94 software. The "itc hi Transmit Event" 94 software will transmit the name of the event and the data or event information associated with that named event to the ITC-Framework (ITC Core) 34 of the particular client application of FIGS. 4 and 5. For example, for a depth event, the depth data and the depth event name will be sent, by the "itc hi Transmit Event" 94 software, to the ITC Framework (ITC Core) 34. For a color event, the new color data and the color event name will be sent, by the "itc hi Transmit Event" 94 software, to the ITC Core 34.

151 In FIG. 29, the actual program code which corresponds to the "Send An Event software" 32b of FIGS. 5 and 28 is illustrated.

152 Referring to FIGS. 30 and 31, a detailed construction and a functional operation of the "Receive An Event Software" 32c of FIG. 5 is illustrated.

153 In FIG. 30, assume that a particular client application sends a plurality of interest objects to the other client applications via the server 26c2, and that one or more of the other client applications will, in response thereto, send the requested events directly to the particular client application via line 40 in FIG. 6. The ITC Framework (otherwise known as the "ITC Core") 34 associated with the particular client application will receive the one or more events from the line 40 of FIG. 6 which originated from the other client applications.

154 Call to Reception Function 96

155 In FIGS. 5 and 30, the ITC Framework (Core) 34 of the particular client application will input the received events (which are received from the other client applications via line 40 of FIG. 6) to the "Receive An Event" software 32c of FIG. 5. The "Receive An Event" software 32c of FIG. 5 includes a block of code which is hereinafter called the "Call to Reception Function" 96 code.

156 Recall from FIGS. 26 and 26A that the block 80, stored in the ITC HI Setup 32a software of FIG. 5 of a particular client application and called the "Build List of ITC Events", stored a list of events, a list of functions corresponding respectively to the list of events, and a list of interest objects corresponding respectively to the list of events and the list of functions. The "Call to Reception function" 96 code of the Receive An Event 32c software of FIGS. 5 and 30 will compare the received events (received from the other client applications via the ITC Core 34) with the plurality of events listed in the "Build List of ITC Events" 80 stored in the ITC HI Setup software 32a of the particular client application, and, when one or more matches are found between a received event and an event stored in the "Build List of ITC Events" block 80, the "Call to Reception function" 96 code will cause the particular functions (82, 84 of FIG. 26) associated with the matched events to be executed by the processor (24a, 26a of FIG. 3) of the particular client application. In FIG. 30, when the functions associated with the matched events are executed by the processor of the particular client application, the particular client application will react accordingly, as indicated by the "Application to React"

block 98 in FIG. 30; that is, the functions will be displayed on the window 12b of the display screen 12a.

157 In FIG. 31, the actual program code which corresponds to the "Receive An Event software" 32c of FIGS. 5 and 30 is illustrated.

158 Referring to FIG. 32, an Intertask Communication (ITC) Sessions Model is illustrated. In FIG. 1, a workstation 10 is illustrated having a screen display 12a which shows a plurality of different windows 12b. Since each window 12brepresents a different client application program 10 executing in the workstation, a single workstation 10 can therefore simultaneously execute a plurality of different client application programs 20. In FIG. 2, the plurality of different windows 12b or client application programs 20 displayed on the display screen 12a could include or consist of a plurality of different client applications 20, such as modelling or Cross View or MapView or 3D View or Seismic or Well View or ELAN or Litho or Bor View.

159 FIG. 32 illustrates the plurality of different client applications 20 executing in the workstation 10. For example, in FIG. 32, a first client application 100, a second client application 102, and a third client application 104 can execute concurrently in the workstation 10 of FIG. 1. An application program data manager 106 manages the concurrently executing client applications 100, 102, 104. The client applications 100, 102, 104 can listen (108) for interest objects received from another client application via the server 26c2, and, when the interest objects associated with a particular event is received by the client applications 100, 102, 104, the client applications 100, 102, 104 will send (110) the particular event directly to the other client application (but not by way of the server).

160 A functional description of the operation of the Distributed Framework Method and Apparatus of the present invention for Intertask Communication between Workstation Applications is set forth in the following paragraphs with reference to FIGS. 1 through 31 of the drawings.

161 Assume that a plurality of workstations, similar to the workstation 10 of FIG. 1, are interconnected together in the manner shown in FIG. 2. Each workstation 10 of the plurality has at least one window display 12b presented to the operator on the display screen 12a of the workstation 10. Each window display 12b on each workstation 10 is being generated by the Operator Interaction Display software 32d of FIG. 5 of a "client application program" (otherwise known as a "client application") and each client application may present to an operator, sitting at the workstation 10, a different functional representation. For example, as shown in FIG. 2, one client application 20 may present to the operator at the workstation 10 a modelling functional representation, another client application 20 may present to the operator a Cross View functional representation, and another may present to the operator either a MapView or a 3D View or a Seismic or a Well View or an ELAN or a Litho or a Bor View functional representation. Therefore, as indicated in FIG. 2, a plurality of different client applications 20 are interconnected together by the "Distributed Framework" method and apparatus of the present invention adapted for providing an intertask communication between workstation applications.

162 One of the workstations 26 of FIG. 3, representing one client application 20 of FIG. 2, stores the server 26c2 as well as its own particular client application 26c1 as shown in FIG. 3, and the other workstation 24, representing another client application 20 of FIG. 2, stores its own particular client application 24c1 of FIG. 3.

163 Assume that an operator at workstation 24 of FIG. 3 is viewing the log chart 12b shown in FIG. 16 on a window display 12b of the display screen 12a of FIG. 1, the log chart 12b of FIG. 16 including the closed state status icon 60b, the broadcast icon 62, and the event filter icon 64 appearing on the bottom right hand corner of the log chart 12b. The operator does not click "on" the closed

state icon 60b, and the operator does not click "on" either the broadcast icon 62 or the event filter icon 64. As a result, the operator's "door is open"; that is, all events previously requested from other client applications will be received by the log chart 12b client application from other client applications, and all events created by the operator on the log chart 12b client application, which were previously requested by other client applications, will be sent by the log chart 12b client application to the other interested client applications.

164 As a result, when the window display 12b, on the display screen 12a of the workstation 24 of FIG. 3 displaying the log chart 12b client application of FIG. 16, is called up by the operator, the operator interaction display software 32d of FIG. 5 will: (1) display the log chart 12b client application of FIG. 16 in the window 12b of the display screen 12a of the workstation 24 of FIG. 3, and (2) instruct the "Call to itc.sub.-- hi.sub.-- Filter and Session" 86 of FIG. 26 of the ITC-HI Setup software 32a of FIG. 5 to send the interest objects 80b of FIG. 26A, associated with the plurality of events 80a in the "list of ITC events" 80 in the ITC HI Setup software 32a of FIG. 5, to the Send An Event software 32b of FIG. 5. The Send An Event software 32b will, in turn, send the interest objects 80b to the ITC Framework 34, of the log chart 12b client application 24c1, of FIG. 5. The ITC Framework 34 of the log chart 12b client application 24c1 will send the interest objects 80b to the server 26c2 via line 36 of FIG. 6. The server 26c2 will register the interest objects therein and will send the interest objects to all other client applications 20 of FIG. 2 including the client application 2 26c1 shown in FIG. 6. The ITC Framework 34 of the client application 2 26c1 will send the received interest objects to the Receive An Event software 32c of FIG. 5, which will, in turn, send the received interest objects to the "Call to itc.sub.-- hi.sub.-- Filter And Session" 86 of FIG. 26 of the ITC HI Setup Software 32a of FIG. 5 of the client application 2 26c1. The "Call to itc hi filter And Session" 86 of client application 2 26c1 will compare the received interest objects with the interest objects 80b stored in the "Build List of ITC Events" 80 of FIGS. 26 and 26A of client application 2. When a match is found between a received interest object and one of the interest objects 80b of FIG. 26A for client application 2 corresponding to a particular event, such as "event N", the "Call to itc.sub.-- hi.sub.--

165 Filter and Session" 86 will send the "event N" to the Send An Event software 32b of FIG. 5 of the client application 2 26c1 which will, in turn, send the "event N" to the ITC Framework 34 of the client application 2 26c1 of FIG. 5. The ITC Framework 34 for client application 2 26c1 of FIG. 5 will send the "event N" directly to the log chart client application 24c1 of FIG. 6 via line 40 of FIG. 6 without requiring the "event N" to register with and pass through the intervening server 26c2.

166 Assume now that the operator at the workstation 24 of FIG. 3, viewing the log chart 12b client application of FIG. 16 on the window 12b of the display screen 12a, clicks "on" the event filter icon 64 in FIG. 16. The "clicking on" of the event filter icon 64 in FIG. 16 will call up the event filter subwindow 64a in FIGS. 15c, 15d, and 15e. The subwindow 64a will have a plurality of events listed therein, the plurality of events consisting of the events (event 1, event 2, event 3, . . . , and event N) shown in FIG. 26A.

167 In FIG. 15e, assume that the operator clicks "on" the "all" portion 64a 4 and 64a5 in the "send" and "receive" column 64a1 and 64a2 of the event filter subwindow 64a. As a result, when the log chart 12b client application 24c1 sends the interest objects 80b of FIG. 26A to the server 26c2 of FIG. 6 and the server 26c2 sends the interest objects to the client application 2 26c1, the client application 2 will send event information associated with any one or all of the events 1, . . . , event N directly to the client application 1 via line 40 of FIG. 6 and the client application 1 24c1 will receive all of the events.

168 Conversely, when the client application 2 26c1 sends the interest objects 80b of FIG. 26A to the server 26c2 of FIG. 6 and the server 26c2 sends the interest

objects to the log chart client application 1 24c1, the client application 1 will send event information associated with any one or all of the events 1, . . . , event N directly to the client application 2 via line 40 of FIG. 6 and the client application 2 26c1 will receive all of the events.

169 However, assume that the operator viewing the subwindow 64a of the event filter icon 64 of FIG. 15e (of the log chart 12b client application 24c1 of FIGS. 3 and 16 on the workstation 24 of FIG. 3) clicks "send" (1A1 of FIG. 15e) but not "receive" (2A1 of FIG. 15e) for event 1, but clicks both "send" (1A2, 1A3, 1A4) and "receive" (2A2, 2A3, and 2A4) for all other events, event 1, event 2, . . . , and event N in the event filter icon subwindow 64a of FIG. 15e. The log chart client application 24c1 will send the event 1 to the client application 2 26c1 of FIG. 3 via line 40 of FIG. 6 (when the client application 2 requested the event 1 from the log chart client application 1 via the server), but the log chart client application 24c1 will not receive the event 1 from the client application 2 26c1 of FIG. 3 via line 40 of FIG. 6 (when the log chart client application 1 requested the event 1 from the client application 2 via the server). However, all other events, event 2, event 3, . . . , and event N, will be received from client application 2 by the log chart client application 24c1 and will be sent to the client application 2 by the log chart client application 24c1.

170 Part 2--Integrated Data Communication and Data Access System including the Application Data Interface

171 Background

172 In a prior pending application Ser. No. 08/758,833 filed Dec. 4, 1996 and entitled "Distributed Framework for Intertask Communication Between Workstation Applications" to Shamim Ahmed and Serge J. Dacic (termed the "ITC application"), a Distributed Framework method and apparatus was disclosed for providing direct inter-task communications (ITC) between concurrently operating computer program applications executing in one or more computer workstations that provide a window display to an operator. The aforementioned "ITC application" was discussed above and has already been incorporated by reference into the specification of this application.

173 Although not disclosed in the "ITC Application", the above referenced "ITC Application" utilizes an underlying apparatus which interfaces with a Database, and that underlying apparatus is called the "Application Data Interface" or "ADI". The "Application Data Interface (ADI)" is embodied in a system called the "Integrated Data Communication and Data Access System" which is the subject matter of the invention of this application.

174 The remaining portion of this specification discloses in detail the "Integrated Data Communication and Data Access System" of the present invention and it is comprised of two parts: a "General Description" which discloses the broader concepts of the Integrated Data Communication and Data Access System of the present invention, and a "Detailed Description" which discloses the more detailed aspects of the Integrated Data Communication and Data Access System.

175 Recall from FIGS. 6 through 9B that a first client application will notify a server when the first client application is interested in receiving event information from another second client application. The server will then further notify the second client application regarding the first client application's interest in the event information. When the second client application practices an event which produces that event information, the second client application will send that event information directly to the first client application without first registering that event information with the server. For example, consider the following FIGS. 33 through 35 which will serve as a review of the above referenced concept.

176 Referring to FIGS. 33 through 35, the drawings of FIGS. 6, 8A, and 9A are again

illustrated.

177 In FIG. 33, a client application 1 24c1 sends an interest object, via line 36, to the server 26c2 requesting to receive certain event information when client application 2 26c1 practices an event which produces that event information. The server 26c2 will forward that request, via line 38, directly to client application 2 26c1. When client application 2 26c1 practices an event which produces that requested event information, the client application 2 26c1 will send the requested event information, via line 40, directly to client application 1 24c1 without registering the requested event information with the server 26c2.

178 In FIG. 34, the application 1 (App 1) 24c1 requests that event information by sending the interest object, via line 46, to the server 26c2, and the server 26c2 forwards that interest object to application 2 (App 2) 26c1 via line 48, application 3 (App 3) 42 via line 50, and application 4 (App 4) 44 via line 52.

179 In FIG. 35, when "App 2" 26c1 practices an event which produces that event information, "App 2" 26c1 transmits the requested event information directly to "App 1" 24c1 without first registering that event information with the server 26c2 (the server remains idle).

180 Referring to FIGS. 36 through 76, the "Integrated Data Communication and Data Access System" in accordance with the present invention is illustrated.

181 In FIGS. 45 through 76, the "Detailed Description" set forth below refers to FIGS. 45 through 76 and provides a detailed discussion of the "Application Data Interface (ADI)" which is embodied in the Integrated Data Communication and Data Access System.

182 In FIGS. 36 through 44, however, the "General Description" set forth below refers to FIGS. 36 through 44 and provides a general discussion of the concepts associated with the "Integrated Data Communication and Data Access System" of the present invention which includes the ADI of FIGS. 45-76.

183 General Description

184 Refer now to FIGS. 36 through 39.

185 In FIG. 36, an "Integrated Data Communication and Data Access System" is illustrated which includes an "Application Data Interface" or "ADI" 115 that is operatively interconnected between a data store or database 110, an Application A (or first client application) 24c1, and an Application B (or second client application) 26c1. The Application data interface 115 will write a "dataItem X" to the database 110, and it executes a callback function in Application B 26c1.

186 In FIG. 37, a further construction of the "Integrated Data Communication and Data Access System" of FIG. 36 is illustrated. In particular, the Application Data Interface (ADI), embodied in the system of FIG. 37, is discussed below in terms of the function the "ADI" performs in the system of FIG. 37 relative to the transfer and inter-communication of originally created and subsequently modified data between a first client application and its cache memory, a database, a server, and a second client application and its cache memory.

187 In FIG. 37, the Integrated Data Communication and Data Access System of FIG. 36 includes first client application 24c1 is operatively connected the server 26c2 via an operative connection 36, as before. The first client application 24c1 is also operatively connected to the second client application 26c1 via the operative connection 40. The second client application 26c1 is also operatively connected to the server 26c2 via the operative connection 38. The first client application 24c1 includes an "Application 1" 111, which communicates with the server 26c2, and a cache memory (cache 1) 119 operatively connected to the "Application 1" via the Application Data Interface (ADI) 115. The "cache 1" 119

stores a Data Object 119 that is generated by the "Application 1". The "cache 1" 119 which stores the Data Object 119 responds to a "create" command, a "delete" command, a "put" command, a "get" command, and a "select" command originating from "Application 1". The second client application 26c1 also includes an "Application 2" 117 which communicates with the server 26c2 and a cache memory (cache 2) 121 operatively connected to the "Application 2" via the Application Data Interface (ADI). The "cache 2" 121 stores a Data Object 121 that is generated by the "Application 2" (usually, the Data Object in "cache 2" is the same as the Data Object in "cache 1"). The "cache 2" 121 which stores the Data Object also responds to a "create" command, a "delete" command, a "put" command, a "get" command, and a "select" command originating from "Application 2". These commands will be discussed later. The "Application 1" is operatively connected to a database 110 via two operative connections (connection 112 and connection 114) for the purpose of storing data in the database 110 when "cache 1" is set in a transient state or when "cache 1" is set in a persistent state. The "Application 2" is also operatively connected to the database 110 via two operative connections (connection 116 and connection 118) for the purpose of storing data in the database 110 when "cache 2" is set in a transient state or when "cache 2" is set in a persistent state. When "Application 1" stores data in the database 110 during the transient state, it will do so via operative connection 112; however, when "Application 1" stores data in the database 110 during the persistant state, it will do so via operative connection 114. Similarly, when "Application 2" stores data in the database 110 during the transient state, it will do so via operative connection 116; however, when "Application 2" stores data in the database 110 during the persistant state, it will do so via operative connection 118. The terms "transient" and "persistent" will be defined later in this specification. The database 110 stores a plurality of Data Objects, including a Data Object 110a. Usually, the Data Object 110a in the Database 110 is the same as the Data Object in "cache 1and the Data Object in "cache 2". In response to the "put" storage state (ss) command originating from "Application 1" and received by the "cache 1" 119 which stores the Data Object 119, the "cache 1" of the first client application 24c1 will be set in one of three separate storage states (ss): the "persistent" storage state or the "transient" storage state or the "memory" storage state. When the "cache 1" is set in one of these storage states, the "cache 2" will be automatically set in the same storage state as that of "cache 1". Similarly, in response to the "put" command originating from "Application 2" and received by the "cache 2" 121 which stores the Data Object 121, the "cache 2" of the second client application 26c1 will be set in one of the three above referenced separate storage states (ss): the "persistent" storage state or the "transient" storage state or the "memory" storage state. When the "cache 2" is set in one of these storage states, the "cache 1" will also be automatically set in the same storage state as that of "cache 2".

188 In FIG. 37, as noted earlier, the cache memories 119, 121 which store the first and second Data Objects 119, 121 are each adapted to receive either a "create" command, or a "delete" command, or a "put" command or a "get" command or a "select" command from "Application 1" and "Application 2", respectively. When the cache memories 119, 121 storing the Data Objects 119, 121 receive any one of these commands, the "cache 1" or the "cache 2" will respond accordingly. For example, when the cache memories 119, 121 storing the Data Objects 119, 121 each receive the "put" storage state command from "Application 1" and "Application 2" respectively, the "cache 1" or "cache 2" will be set in either the "persistent" storage state or the "transient" storage state or the "memory" storage state; and, after the appropriate "storage state" is set, the Data Objects 119, 121 can then be modified or changed by "Application 1" or "Application 2".

189 When, in response to the "put" command, the "Application 1" sets the "persistent" storage state, and when the "Application 1" subsequently creates an "original set of data" and then subsequently modifies or changes the original set of data to create "modified data", both the "original set of data" and the "modified data" will be stored in the database 110 via the operative connection 114 (because the "persistant" storage state has been set). On the

other hand, when, in response to the "put" command, the "Application 1" sets the "transient" storage state, and when the "Application 1" subsequently creates the "original set of data" and then subsequently creates the "modified data", the "original set of data" will be stored in the database 110 via the operative connection 112, however, the "modified data" will not be stored in the database 110 (because the "transient" storage state has been set). On the other hand, when, in response to the "put" command, the "Application 1" sets "memory" storage state, and when the "Application 1" subsequently creates the "original set of data" and then subsequently creates the "modified data", none of the "original set of data" and none of the "modified data" will be stored in the database 110 (because the "memory" storage state has been set).

190 Similarly, when, in response to the "put" command, the "Application 2" sets the "persistent" storage state, and when the "Application 2" subsequently creates an "original set of data" and then subsequently modifies or changes the original set of data to create "modified data", both the "original set of data" and the "modified data" will be stored in the database 110 via the operative connection 118 (because the "persistent" storage state has been set). On the other hand, when, in response to the "put" command, the "Application 2" sets the "transient" storage state, and when the "Application 2" subsequently creates the "original set of data" and then subsequently creates the "modified data", the "original set of data" will be stored in the database 110 via the operative connection 116, however, the "modified data" will not be stored in the database 110 (because the "transient" storage state has been set). On the other hand, when, in response to the "put" command, the "Application 2" sets "memory" storage state, and when the "Application 2" subsequently creates the "original set of data" and then subsequently creates the "modified data", none of the "original set of data" and none of the "modified data" will be stored in the database 110 (because the "memory" storage state has been set). When the cache memory 119 or 121 storing the Data Object 119 or 121 receives the "create" command, and the "persistant" or "transient" storage state has been set, the "Application 1" or "Application 2" will create and store another Data Object 110a in the data base 110. When the cache memory 119 or 121 storing the Data Object 119 or 121 receives the "select" command, the "Application 1" or "Application 2" will select a Data Object 110a stored in the data base 110, and when the cache memory 119, 121 storing the Data Object 119 or 121 receives the "get" command, the "Application 1" or the "Application 2" will retrieve the selected Data Object 110a from the data base 110. When the cache memory 119, 121 storing the Data Objects 119 or 121 receives the "delete" command, the

191 "Application 1" or the "Application 2" will delete the Data Object 110a from the data base 110.

192 In FIG. 38a and 38b, block diagrams are presented which illustrate the definitions of the "transient" data transfer and the "persistent" data transfer.

193 In FIG. 38a, the definition of "transient" data transfer" is illustrated. In FIG. 38a, an application data interface 115 assists the transfer of data from a writer application 111 and a temporary cache memory 119. Since, in FIG. 38a, we are illustrating the function of the "transient" data transfer, the original, initially created data currently residing in the temporary memory 119 will be written into the database storage 110; however, because we are in the "transient" data transfer mode, no further corresponding modified data will be written into the database storage 110. In FIG. 38a, the ADI 115 will also assist the reading of the initially stored data from the database storage 110 to another temporary cache memory 121, and from the temporary memory 121 to the reader application 117 for the modified data via operative connection 40.

194 In FIG. 38b, the definition of the "persistent" data transfer is illustrated. In FIG. 38b, an application data interface 115 also assists the transfer of data between a writer application 111 and a temporary cache memory 119. Since, in FIG. 38b, we are illustrating the function of the "persistent" data transfer, the original, initially created data currently residing in the

temporary cache memory 119 will be written into the database storage 110; however, because we are in the "persistent" data transfer mode, all further corresponding modified versions of the data will also be written into the database storage 110. In FIG. 38b, the ADI 115 will also assist the reading of the initially stored data from the database storage 110 to another temporary cache memory 121, and from the temporary cache memory 121 to the reader application 117.

195 In FIG. 39, a flow diagram is illustrated which describes the functional operation of the interactive data communication system of FIGS. 36 and 37. In particular, the diagram of FIG. 39 describes the functional operation of the "Application Data Interface (ADI)" inherently embodied in the system of FIG. 37 which governs the transfer and inter-communication of originally created and subsequently modified data throughout the system of FIG. 37. In FIG. 39, the following functional steps are performed by the system of FIG. 37, which steps are governed by the Application Data Interface of the present invention:

196 (1) App 1 (111) creates a Data Object 110a in a cache memory, while in the transient or persistant mode, and then stores the newly-created Data Object in the database 110, block 124,

197 (2) App 2 (117) queries/reads the Data Object 110a from the database 110, block 126,

198 (3) App 2 (117) sends an interest object to the server 26c2, block 128,

199 (4) the server 26c2 registers the App 2 interest in the Data Object 110a and distributes the interest object to App 1 (111), block 130,

200 (5) App 1 (111) generates a Data Object for Event "X", block 134,

201 (6) App 1 (111) sends a notification to App 2 (117) and updates the Data Object 110a in the database 110 when App 1 is in the persistent mode, block 136, and

202 (7) App 1 sends an updated Data Object for Event "X" directly to App 2, via line 40, without registering the updated Data Object with the server, block 138.

203 Each of these steps in the flow diagram of FIG. 39 will be discussed in detail in the following functional description of the operation of the Integrated Data Communication and Data Access System of FIG. 37, the sequential flow of data through such System being governed by the Application Data Interface (ADI) of the present invention.

204 In FIG. 37, a functional description of the operation of the "Integrated Data Communication and Data Access System" of FIG. 37, including the Application Data Interface 115 embodied therein, will be set forth in the following paragraphs with reference to FIG. 37.

205 In FIG. 37, assume that the "Application 1" of the first client application 24c1 has set the "memory" storage state. While in the "memory" storage state, the "Application 1" creates "new data", stores that new data in the cache memory 119 in the form of a Data Object 119, changes/modifies that new data thereby creating "modified data", re-stores the modified data in the cache memory 119 in the form of a new Data Object 119, continuously modifies the "modified data" thereby creating "further modified data", and restores the further modified data in the cache memory 119 in the form of a further new Data Object 119. Since this function occurred after the "Application 1" set the "memory" storage state, none of the newly or subsequently created data was stored as a Data Object 110a in the database 110, that is, the new data, the modified data, and the further modified data was not stored as a Data Object 110a in the database 110.

206 Assume now that the "Application 1" sets the persistent or transient storage state by generating the "put" storage state command which is received by the cache memory 119. Therefore, the cache memory 119 is now set in the persistant or transient storage state. When the "persistant" or the "transient" storage state is set by "Application 1", assume that "Application 1" practices an event (called "event X") and thereby generates a set of "original and newly created data". When the "original and newly created data" is generated by "Application 1", since the "persistent" or "transient" storage state has been set, the ADI 115 will respond to the "Application 1" by storing the "original and newly created data" in the form of the Data Object 119 in the cache memory 119 of the first client application 24c1; and, in addition, the ADI 115 will also respond to the "Application 1" by storing the "original and newly created data" in the form of the Data Object 110a in the Database 110. Therefore, the Data Objects 119 and 110a were both created in response to and as a direct result of the function of the ADI 115 responsive to the practice of event X by "Application 1" during the setting of the "persistant" or "transient" storage state. However, if the "memory" storage state had been set, the ADI 115 would have stored the "original and newly created data" in the cache memory 119 in the form of the Data Object 119; but the ADI 115 would not have stored the "original and newly created data" in Database 110 in the form of Data Object 110a.

207 If the "Application 1" set the persistant storage state, the "cache 1" and the "cache 2" are both set in the persistant storage state. As a result, the operative connection 118 indicative of the "persistent" storage state is now open and the "Application 2" of the second client application 26c1 can update the Data Object 110a in the database 110 whenever that Data Object 110a is changed or modified by the "Application 2". The "Application 2" of the second client application 26c1 queries the database 110, via operative connection 118, and notices that the Data Object 110a (which was previously stored in the database 110 by "Application 1") exists and is stored in the database 110. When the "Application 2" notices that Data Object 110a, the "Application 2" of the second client application 26c1 becomes interested in receiving more data associated with that Data Object 110a from "Application 1" of the first client application 24c1 whenever "Application 1" again practices or continues to practice the "event X"which created the Data Object 110a in the database 110. As a result, the "Application 2" of the second client application 26c1 sends an interest object to the server 26c2 via operative connection 38. The server 26c2 registers that interest object from "Application 2" and forwards that interest object to the "Application 1" of the first client application 24c1 via operative connection 36. Whenever "Application 1" of the first client application 24c1 again practices the "event X" thereby creating "newly updated data", the "newly updated data" associated with the practice of "event X" will be transmitted directly from "Application 1" to "Application 2" via the operative connection 40 in FIG. 37 without passing through and registering with the server 26c2.

208 Assume now that the "Application 1" of the first client application 24c1 sets the "persistent" storage state by generating the "put" storage state command which is received by the cache memory 119. As a result, the" cache 1" and the "cache 2" are both set in the "persistent" storage state. Assume now that the "Application 1" of the first client application 24c1 begins to practice the "event X". During the practice of "event X" by "Application 1", some "original and newly created data" is generated by the "Application 1" and some "further subsequently modified data" is subsequently generated by the "Application 1". In other words, during the practice of "event X" by "Application 1", the data resultant from the practice of event X is constantly changing. Since the "Application 1" of the first client application 24c1 is set in the "persistent" storage state, the ADI 115 will respond to the "Application 1" by storing the "original and newly created data" as Data Object 110a in the Database 110 via operative connection 114, and the ADI will further respond to the "Application 1" by further storing the "further subsequently modified data" as Data Object 110a in the Database 110 via the operative connection 114.

209 Assume now that the "Application 1" of the first client application 24c1 sets the "transient" storage state by generating the "put" storage state command which is received by the cache memory 119 which stores the Data Object 119. As a result, the "cache 1" and the "cache 2" are both set in the "transient" storage state. Since the "cache 1" is set in the "transient," storage state, whenever any "original and newly created data" is generated by "Application 1" as a result of the practice by "Application 1" of "event X", the ADI 115 will respond to the "Application 1" by storing that "original and newly created data" as Data Object 110a in the Database 110 via the operative connection 112. However, during the continual practice of "event X" by "Application 1", further "modified and subsequently created data" will be generated by "Application 1". Since the "cache 1" is set in the "transient" storage state, the ADI 115 will respond to the "Application 1" by not storing any of the "modified and subsequently created data" as Data Object 110a in the Database 110.

210 Assume now that the "Application 1" sets the "memory" storage state by generating the "put" storage state command which is received by the cache memory 119. Since the "cache 1" is set in the "memory" storage state, if any "original, newly created data" is produced by the "Application 1", and if any "further, subsequently modified data" is produced by "Application 1", none of the "original, newly created data" and none of the "further, subsequently modified data" will be stored in the Database 110 as a Data Object 110a.

211 Refer now to FIGS. 40 through 44.

212 In FIG. 40, a port 142 is operatively connected to a cache memory 144 (or it can be a buffer memory) which stores a Data Object 144, and the Data Object 144 is operatively connected to a database 110. The port 142 includes a converter 142a. The port 142 responds to a read command, at terminal 142b and a write command, at terminal 142c. The port 142 is an opaque handle that can be used to access associated structures. It is a special type of file handle where the handle itself contains all the information to access the data. The port 142 is discussed in detail in the "Detailed Description of the Preferred Embodiment" set forth below. The port 142 is always in the "memory" storage state, as indicated by numeral 146 in FIG. 40.

213 In FIG. 41, the port 142 includes a converter 142a. The port 142 is adapted to transfer data between an output terminal 148 (which operatively interconnects the port 142 to the cache 144 which stores the Data Object 144) and either the read command terminal 142b or the write command terminal 142c. When the data is being transferred between the output terminal 148 and either the read command terminal 142b or the write command terminal 142c, the data undergoes a conversion process within the converter 142a. For example, if the data is being transferred from the output terminal 148 and to the read command terminal 142b, the data is converted, by the converter 142a, from a format A (at terminal 148) to a format B (at terminal 142b); however, if the data is being transferred from the write command terminal 142c to the output terminal 148, the data is reconverted back, by the converter 142a, from the format B (at terminal 142c) to the format A (at terminal 148). For example, the data may be converted from a metric unit of measure to an English or Canadian unit of measure. The significance of this function will become evident from a reading of the following functional description of the Integrated Data Communication and Data Access System of the present invention, which includes the Application Data Interface 115, with reference to FIGS. 42 through 44.

214 In FIG. 42, a more detailed construction of the converter 142a of the Port 142 of FIG. 41 is illustrated. In FIG. 42, when the data is transferred between the output terminal 148 and either the read command terminal 142b or the write command terminal 142c, the data will pass through a plurality of conversion units, as follows: the data will pass through a gate conversion unit 142a1, a filter conversion unit 142a2, a coordinate conversion unit 142a3, a units value type conversion unit 142a4, and/or a user conversion unit 142a5. Each of these conversion units 142a1 through 142a5 are discussed in detail in the "Detailed Description of the Preferred Embodiment" set forth below.

215 In FIG. 43, by way of example, data from a "data domain" is being written into the port 142 via the write command terminal 142c. When that data is written into the port 142 via the write command terminal 142c, that data undergoes a first conversion in the user defined gate conversion unit 142a1; then, the data undergoes a second conversion in the filter conversion unit 142a2; then, the data undergoes a third conversion in the units value type conversion unit 142a4; and then the data undergoes a fourth conversion in the user conversion unit 142a5. After the data undergoes conversion in the user conversion unit 142a5, the converted data passes to the cache 144 which stores the Data Object 144 via the output terminal 148. An example of a typical conversion executed by the converter 142a in port 142, similar to the conversion described above in FIGS. 42 and 43, will be provided below with reference to FIG. 44 of the drawings.

216 In FIG. 44, the Integrated Data Communication and Data Access System of FIG. 37 is again illustrated in FIG. 44; however, in FIG. 44, the first client application 24c1 and the second client application 26c1 of the Integrated Data Communication and Data Access System each further include the apparatus of FIG. 40, that is, the port 142 including the converter 142a operatively connected to the cache 119, 121 which stores the Data Object 119, 121. For example, the first client application 24c1 includes a "port 1" 142 (including a "converter 1" 142a) operatively connected to the cache memory (cache 1) 119 which stores the Data Object 119, and the second client application 26c1 includes a "port 2" 142 (including a "converter 2" 142a) operatively connected to the cache memory (cache 2) 121 which stores the Data Object 121. In addition, a "function" 120 is associated with the "cache 1" 119, and a "function" 122 is associated with the "cache 2" 121. The purpose of the "function" 120, 122 in the Integrated Data Communication and Data Access System of FIG. 44 will become clear following a reading of the following functional description of the operation of the present invention.

217 A functional description of the operation of the Integrated Data Communication and Data Access System of FIG. 44, including the function of the Application Data Interface (ADI) 115, is set forth in the following paragraphs with reference to FIG. 44.

218 In FIG. 44, assume that the first client application 24c1 is located in a first location on earth having a first system of data measure (e.g., the English unit of measure) and the second client application 26c1 is located in a second location on earth having a second system of data measure (e.g., the Canadian unit of measure). The first client application 24c1 is a windowed application program being presented to a first operator sitting at a first workstation located in the first location on earth, and the second client application 26c1 is another windowed application program being presented to a second operator sitting at a second workstation located in the second location on earth. The "Application 1" 111 of the first client application 24c1 represents a particular program application that is being executed; and, during the execution of "Application 1", a

219 certain type of data is required from the first operator viewing the first client application 24c1. During the execution of the "Application 1", an "event" is being practiced by the "Application 1". In order to provide the required certain type of data, the first operator viewing the first client application 24c1 provides that certain type of data by writing a "first type of data having the English units of measure", via the write command terminal 142c, to the "port 1" 142 of the first client application 24c1. The converter 142aof the "port 1" 142 will receive the "first type of data having the English units of measure" and it will convert the "first type of data having the English units of measure" into the "first type of data having a metric unit of measure". The "port 1" 142 in the first client application 24c1 has already been set in the "memory" storage state in response to the "put" storage state command received by the "Port 1" 142. The "first type of data having the metric units of measure", that is being output from the "port 1" 142, is input to the

"cache 1" 119 and is stored as a Data Object 119 in the "cache 1" of the first client application 24c1. Assume that the "cache 1" was previously set in the persistant storage state. As a result, the "first type of data having the metric units of measure", which is temporarily stored in the "cache 1" 119 as Data Object 119 of the first client application 24c1, can now be transferred from the "cache 1" 119 to the database 110, via the "persistent" operative connection 114, and stored therein in the form of a Data Object 110a.

220 As a result, the "Application 1" practiced an "event", and, when that "event" was practiced by "Application 1", certain "event information" was generated by "Application 1". That "event information" was stored in the database 110 in the form of the Data Object 110a, the Data Object 110a representing the aforementioned "first type of data having the metric units of measure".

221 The second operator viewing the second client application 26c1 reads the Data Object 110a (representing the "first type of data having the metric units of measure") by issuing a read command via the read command terminal 142b of the "Port 2" of the second client application 26c1. As a result, the "first type of data having the metric units of measure", stored as the Data Object 110a of the database 110, is transferred from the database 110 and to the "cache 2" 121, via the "persistent" operative connection 118, and is temporarily stored in the "cache 2" 121 as the Data Object 121 of the second client application 26c1. This transfer of the "first type of data having the metric units of measure" from the database 110 to "cache 2" can occur because the "Application 2" 117 has already set the "persistant" storage state in response to the "put" command received in the "cache 2" 121. The "first type of data having the metric units of measure", now stored in the "cache 2" 121 as Data Object 121 of the second client application 26c1, is transferred from the "cache 2" 121 to the "port 2" 142 of the second client application 26c1. The converter 142a of the "port 2" converts the received "first type of data having the metric units of measure" into the "first type of data having the Canadian units of measure". When the converter 142a converts the "first type of data having the metric units of measure" into the "first type of data having the Canadian units of measure", the second operator viewing the second client application 26c1 can now read the "first type of data having the Canadian units of measure" by issuing a read command via the read command terminal 142b of the "port 2" 142 of the second client application 26c1. Assume that, when the second operator of the second client application 26c1 views (on his display screen of his workstation) the "first type of data having the Canadian units of measure", the second operator of the second client application 26c1 becomes interested in receiving more "subsequently created and updated data" that is associated with the "first type of data having the Canadian units of measure". As a result of this increased interest in the "subsequently created and updated data" on the part of the second operator of the second client application 26c1, the second operator will express that interest by writing an "interest object" from the write terminal 142c to the "port 2" 142 of the second client application 26c1. That "interest object" is not converted by the converter 142a; rather, the "interest object" passes directly to the "cache 2" 121. When the "interest object" is received by the "cache 2" 121, the "function" 122 is set. Now that the "function" 122 is set, when the "subsequently created and updated data" is received in the "cache 2" from the first client application 24c1 via operative connection 40, the particular function associated with that "updated data" will be implemented by the "Application 2" 117 and that particular function will be presented to the second operator of the second client application 26c1 for viewing on his display screen on his workstation. The ADI 115 of the second client application 26c1 passes the "interest object" to the "Application 2" 117, whereupon, the "Application 2" 117 of the second client application 26c1 will pass that "interest object" to the Server 26c2 via operative connection 38. The server 26c2 registers the "interest object" received from the second client application 26c1, and then the server 26c2 forwards that "interest object" to the "Application 1" 111 of the first client application 24c1. The ADI 115 of the first client application 24c1 will pass that "interest object" to the "cache 1" 119. When that "interest object" is received by the "port 1" 142, it does not undergo conversion in the converter 142; rather, it is read from the

"port 1" 142 via the read terminal 142 and it is presented to the first operator of the first client application 24c1 for viewing on the first operator's workstation display screen. When that "interest object" is received by the "Application 1" 111, the "interest object" is associated with a particular "event" in the "Build List of ITC Events" 80 of FIG. 26A.

222 Assume now that the "Application 1" 111 re-practices the same "event" which originally generated the "event information" representing the "first type of data having the metric units of measure". Recall that the "first type of data having the metric units of measure" was stored in the database 110 as Data Object 110a. When the "Application 1" re-practices that same "event", the "Application 1" will generate the requested "subsequently created and updated data". Recall that the "Application 2" of the second client application 26c1 has already expressed interest in the "subsequently created and updated data".

223 When the "event" is re-practiced, "Application 1" 111 of the first client application 24c1 is executing, and the execution of "Application 1" requires that certain "updated input data in the English units of measure" be provided by the first operator of the first client application 24c1. The first operator therefore provides that "updated input data in the English units of measure" by writing that data to the "port 1" 142 of the first client application 24c1 via the write terminal 142c. The "updated input data in the English units of measure" provided by the first operator represents the "subsequently created and updated data" that was requested by the second client application 26c1. The converter 142a of the "port 1" converts the "updated input data in the English units of measure" into a "updated input data in the metric units of measure", and the "updated input data in the metric units of measure" is temporarily stored in the "cache 1" 119 in the form of a Data Object 119 of the first client application 24c1. However, at this point, the ADI 115 of the first client application 24c1 transfers this "updated input data in the metric units of measure" from the "cache 1" to the "Application 1", and the "Application 1" transfers this "updated input data in the metric units of measure" directly from "Application 1" to "Application 2" of the second client application 26c1 via operative connection 40 in FIG. 44 without registering that "updated input data" with the server 26c2. The "updated input data in the metric units of measure" is also stored in the database 110 if the "cache 1" was set in the persistant storage state. The "updated input data in the metric units of measure" is received in and temporarily stored in the "cache 2" 121 of the second client application 26c1 in the form of the Data Object 121. The presence of the "updated input data in the metric units of measure" stored in "cache 2" in the form of Data Object 121 of the second client application 26c1 triggers the "function" 122. The "updated input data in the metric units of measure" is transferred from the "cache 2" to the "port 2" 142 of the second client application 26c1; and the converter 142a of the "port 2" 142 will convert the "updated input data in the metric units of measure" into "updated input data in the Canadian units of measure". Since the "function" 122 has been triggered, the "function" 122 will display the "updated input data in the Canadian units of measure" in the window (12b of FIG. 1) of "Application 2" which is being displayed on the second operator's workstation display screen. The second operator viewing the second client application 26c1 can now read the "updated input data in the Canadian units of measure".

224 Detailed Description

225 A detailed discussion of the "Application Data Interface" of the present invention in the context of the "Integrated Data Communication and Data Access System" of the present invention is set forth below in the remaining portion of the "Description of the Preferred Embodiment". ##SPC1##

226 The invention being thus described, it will be obvious that the same may be varied in many ways. Such variations are not to be regarded as a departure from the spirit and scope of the invention, and all such modifications as would be obvious to one skilled in the art are intended to be included within the scope of the following claims.

CLAIMS:

We claim:

1. In an integrated data communication and data access system comprising a first client non-server application including a first cache memory, a second client non-server application including a second cache memory, a database interconnected between the first client non-server application and the second client non-server application, and a server interconnected between the first client non-server application and the second client non-server application, a method of communicating between client applications, comprising the steps of:

(a) generating, by said second client non-server application, an original set of data, said second client non-server application adapted to set a persistent storage state and a transient storage state and a memory storage state, said second client non-server application attempting to store said original set of data in said second cache memory and in said database after said second client non-server application sets said persistent storage state and said transient storage state and said memory storage state;

(b) storing, by said second client non-server application, said original set of data in said second cache memory when said second client non-server application sets said persistent storage state or said transient storage state or said memory storage state,

(c) storing, by said second client non-server application, said original set of data in said database when said second client non-server application sets said persistent storage state or said transient storage state, and not storing, by said second client non-server application, said original set of data in said database when said second client application sets said memory storage state;

(d) retrieving by said first client non-server application said original set of data from said database and storing the retrieved original data in said first cache memory of said first client non-server application;

(e) in response to said retrieved original data stored in said first cache memory of said first client non-server application, expressing interest by said first client non-server application in a subsequently modified version of said original set of data by transmitting an interest object associated with an event from said first client non-server application to said server;

(f) receiving said interest object in said server and retransmitting said interest object from said server to said second client non-server application;

(g) generating by said second client non-server application said subsequently modified version of said original set of data when the second client non-server application practices said event, said second client non-server application notifying said first client non-server application when the subsequently modified data is generated and attempting to store said subsequently modified version of said original set of data in said second cache memory and in said database;

(h) storing, by said second client non-server application, said subsequently modified version of said original set of data in said second cache memory when said second client non-server application sets either said persistent storage state or said transient storage state or said memory storage state;

(i) storing, by said second client non-server application, said subsequently modified version of said original set of data in said database when said second client non-server application sets said persistent storage state, and not storing the subsequently modified data in said database when said second client non-server application sets either said transient storage state or said memory

storage state; and

(j) transmitting, by said second client non-server application, said subsequently modified version of said original set of data associated with said event from said second client non-server application directly to said first client non-server application without routing the subsequently modified data through said server.

2. The method of claim 1, further comprising:

(k) responding by said server to one or more revocation objects received from one or more client non-server applications; and

(l) responding by said server to additional interest objects received from other additional client applications.

3. The method of claim 2, wherein the responding step (l) comprises the steps of:

transmitting said interest object associated with said event from a third client non-server application to said server;

retransmitting said interest object from said server to said second client non-server application; and

transmitting said subsequently modified version of said original set of data associated with said event from said second client non-server application to said first client non-server application and to said third client non-server application without routing said subsequently modified version of said original set of data through said server when said second client non-server application practices said event.

4. The method of claim 2, further comprising the steps of:

(m) transmitting a revocation object from said first client non-server application to said server,

(n) in response thereto, transmitting said revocation object from said server to said second client non-server application; and

(o) in response to said revocation object, un-registering, within said second client non-server application, said interest object of said first client non-server application associated with said event and refraining, by said second client non-server application, from sending said event information associated with said event directly to said first client non-server application when said second client non-server application practices said event.

5. The method of claim 2, wherein said first client non-server application is adapted to terminate its execution, and wherein the responding step (l) comprises the steps of:

responding, by said server, to a revocation object received from said first client non-server application and transmitting said revocation object from said server to said second client non-server application when said first client non-server application terminates its execution; and

in response to said revocation object, un-registering, within said second client non-server application, said interest object of said first client non-server application corresponding to said event and refraining, by said second client non-server application, from transmitting said subsequently modified version of said original set of data associated with said event directly to said first client non-server application when said second client

non-server application practices said event.

6. An integrated data communication and data access system adapted for intercommunicating between client applications, comprising:

a second client non-server application adapted to generate an original set of data, said second client non-server application including a second cache memory and adapted to set either a persistent storage state or a transient storage state or a memory storage state;

a database operatively connected to said second client non-server application,

said second client non-server application storing said original set of data in said second cache memory when said second client non-server application sets either said persistent storage state or said transient storage state or said memory storage state,

said second client non-server application storing said original set of data in said database when said second client non-server application sets either said persistent storage state or said transient storage state, and not storing said original set of data in said database when said second client non-server application sets said memory storage state;

a first client non-server application operatively connected to said database and to said second client non-server application, said first client non-server application retrieving said original set of data from said database and subsequently expressing interest in a subsequently modified version of said original set of data by generating and transmitting an interest object corresponding to an event;

a server operatively interposed between and connected to said first client non-server application and said second client non-server application,

said first client non-server application transmitting said interest object to said server,

said server re-transmitting said interest object to said second client non-server application,

said second client non-server application practicing said event and thereby generating a subsequently modified version of said original set of data in response to the practice of said event, said second client non-server application attempting to store said subsequently modified version of said original set of data in said second cache memory and in said database,

said second client non-server application storing said subsequently modified version of said original set of data in said second cache memory when said second client non-server application sets either said persistent storage state or said transient storage state or said memory storage state,

said second client non-server application storing said subsequently modified version of said original set of data in said database when said second client non-server application sets said persistent storage state, and not storing said subsequently modified version of said original set of data in said database when said second client non-server application sets either said transient storage state or said memory storage state,

said second client non-server application, responsive to said interest object, transmitting said subsequently modified version of said original set of data directly to said first client non-server application without routing the subsequently modified data through said server when said second client non-server application practices said event.

7. The integrated data communication and data access system of claim 6, wherein said server responds to a revocation object from said first client non-server application and transmits said revocation object to said second client non-server application when said first client non-server application terminates its execution, said second client non-server application not transmitting said subsequently modified version of said original set of data to said first client non-server application in response to said revocation object when said second client non-server application practices said event.

8. A data communication and data access system, comprising:

a first client application including,

a first application adapted to set a persistent storage state or a transient storage state or a memory storage state,

a first cache memory operatively connected to said first application and responsive to the storage state set by said first application,

first interface apparatus operatively interposed between said first application and said first cache memory adapted to coordinate a transfer of data between said first application and said first cache memory, and

first conversion apparatus operatively connected to said first cache memory and interfacing between a first operator of said first client application and said first cache memory adapted to receive data having a first format from said first operator and convert said data having said first format to data having a second format for storage in said first cache memory, said first conversion apparatus adapted to convert said data having said second format to said data having said first format for use by said first operator,

a second client application including,

a second application adapted to set a persistent storage state or a transient storage state or a memory storage state,

a second cache memory operatively connected to said second application and responsive to the storage state set by said second application,

second interface apparatus operatively interposed between said second application and said second cache memory adapted to coordinate a transfer of data between said second application and said second cache memory, and

second conversion apparatus operatively connected to said second cache memory and interfacing between a second operator of said second client application and said second cache memory adapted to receive data having a third format from said second operator and convert said data having said third format to said data having said second format for storage in said second cache memory, said second conversion apparatus converting said data having said second format to said data having said third format for use by said second operator;

a server operatively interconnected between the first application and the second application; and

a database operatively interconnected between the first cache memory and the second cache memory,

said first conversion apparatus receiving an original set of said data having said first format from said first operator and converting said original set of data having said first format into an original set of data having said second format,

said first application storing said original set of said data having said second format received from said first conversion apparatus into said first cache memory when said first application sets either said persistent storage state or said transient storage state or said memory storage state,

said first application storing storing said original set of said data having said second format received from said first conversion apparatus into said database when said first application sets either said persistent storage state or said transient storage state, said first application not storing said original set of said data having said second format into said database when said first application sets said memory storage state,

said second client application retrieving said original set of said data having said second format from said database and storing said original set of data having said second format in said second cache memory,

said second conversion apparatus converting said original set of data having said second format into an original set of data having a third format for use by said second operator,

said second operator expressing interest in a subsequently modified version of said original set of data having said third format by generating an interest object corresponding to an event,

said second application transmitting said interest object to said server,

said server re-transmitting said interest object to said first application,

said first application subsequently generating a subsequently modified version of said original set of data having a second format and attempting to store said subsequently modified version of said original set of data having said second format into said first cache memory and into said database,

said first application storing said subsequently modified version of said original set of data having a second format into said first cache memory when said first application sets either said persistent storage state or said transient storage state or said memory storage state,

said first application storing said subsequently modified version of said original set of data having a second format into said database when said first application sets said persistent storage state, but not storing said subsequently modified version of said original set of data having a second format into said database when said first application sets either said transient storage state or said memory storage state,

said first application responding to said interest object received from said server by transmitting said subsequently modified version of said original set of said data having said second format directly to said second application without routing said subsequently modified version of said original set of said data having said second format through said server,

said subsequently modified version of said original set of said data having said second format received by said second application being converted by said second conversion apparatus into a subsequently modified version of said original set of said data having a third format for viewing by said second operator.

9. The data communication and data access system of claim 8, wherein said subsequently modified version of said original set of data having said second format which is received by said second application is stored in said second cache memory when said first application or said second application sets either said memory storage state or said persistent storage state or said transient storage state, said second conversion apparatus converting said subsequently

modified version of said original set of data having said second format into said subsequently modified version of said original set of data having said third format for viewing by said second operator.

10. In a data communication and data access system including a first client application which further includes a first application and a first cache memory and a first conversion unit, a second client application which further includes a second application and a second cache memory and a second conversion unit, a server operatively interconnected between the first application and the second application, and a database operatively interconnected between said first cache memory and said second cache memory, a method of data communication and data access, comprising the steps of:

(a) supplying, by a first operator, original data having a first format to said first conversion unit and converting, in said first conversion unit, said original data having said first format to original data having a second format, said first application and said second application adapted for setting a persistent storage state or a transient storage state or a memory storage state,

(b) storing, by said first application, said original data having said second format in said first cache memory when said first application sets either said persistent storage state or said transient storage state or said memory storage state;

(c) storing, by said first application, said original data having said second format in said database when the first application or the second application sets either the persistent storage state or the transient storage state but not storing said original data having said second format in said database when the first application or the second application sets the memory storage state;

(d) retrieving by said second application said original data having said second format from said database and converting, in said second conversion unit, said original data having said second format into original data having a third format for use by a second operator,

(e) in response to said original data having said third format, expressing interest in a modified version of said original data having said third format by transmitting an interest object from said second application to said server and then re-transmitting said interest object from said server to said first application;

(f) supplying, by said first operator, modified data having a first format to said first conversion unit and converting, in said first conversion unit, said modified data having said first format into modified data having a second format;

(g) storing said modified data having said second format in said first cache memory when said first application or said second application sets said persistent storage state or said transient storage state or said memory storage state,

(h) storing said modified data having said second format in said database when the first application or the second application sets the persistent storage state but not storing said modified data having said second format in said database when the first application or the second application sets either the transient storage state or the memory storage state; and

(i) in response to said interest object retransmitted from said server to said first application during step (e), transmitting said modified data having said second format from said first application directly to said second application without routing said modified data having said second format through said server.

11. The method of claim 10, wherein the retrieving and converting step (d) further comprises the steps of:

(d1) reading said original data having said third format from said second conversion unit and supplying said interest object pertaining to said original data having said third format to said second application.

12. The method of claim 11, further comprising the steps of:

(j) storing, by said second application, said modified data having said second format, in said second cache memory when said first application or said second application sets either said persistent storage state or said transient storage state or said memory storage state; and

(k) converting, in said second conversion unit, said modified data having said second format to modified data having a third format for use by said second operator.